(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 113 407 A2

# (12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
**04.07.2001 Bulletin 2001/27**

(51) Int Cl.[7]: **G07F 7/10**, G07F 17/32

(21) Application number: **00128372.0**

(22) Date of filing: **22.12.2000**

(84) Designated Contracting States:
**AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR**
Designated Extension States:
**AL LT LV MK RO SI**

(30) Priority: **27.12.1999 JP 36914299**

(71) Applicant: **Hitachi, Ltd.**
**Chiyoda-ku, Tokyo 101-8010 (JP)**

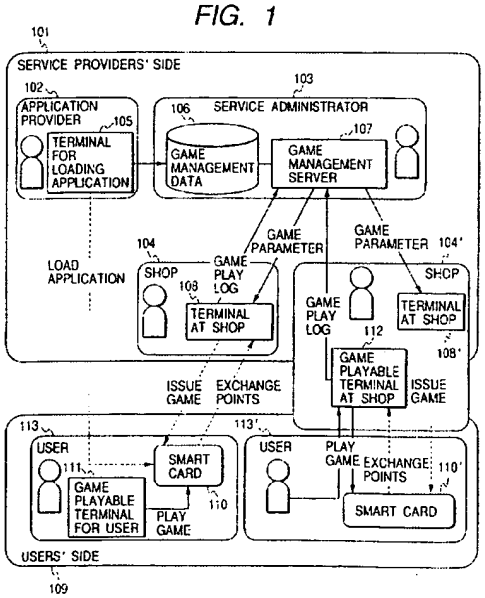
(72) Inventors:
- **Sukeda, Hiroko,**
  **c/oHitachi Ltd. Intel. Prop. Group**
  **, Chiyoda-ku, Tokyo 100-8220 (JP)**
- **Mishina, Yusuke, c/oHitachi Ltd. Int. Prop. Group**
  **, Chiyoda-ku, Tokyo 100-8220 (JP)**
- **Ohki, Masaru, c/oHitachi Ltd. Int. Prop. Group**
  **, Chiyoda-ku, Tokyo 100-8220 (JP)**

(74) Representative: **Strehl Schübel-Hopf & Partner**
**Maximilianstrasse 54**
**80538 München (DE)**

## (54) Smart card, and method of loading application programs and scripts into same

(57) The invention provides a method that enables loading/unloading a plurality of types of games as part of an application program, typically, a game application program installed on a smart card system with high ability of storing information for which highly-reliable security is achievable, extending the use range of the card. Of a program to run on the card, the processing parts that can be executed in common are packaged as modules and game definitions described in scripts are loaded/unloaded into/from the card as required from a terminal operating with the card. In the program, a script interpreter that interprets and executes scripts, a controller that controls scripts loading/unloading, a controller that performs the management of point data and rights to play game are prepared, whereby dynamic loading/unloading of types of games is possible and one application can offer a plurality of types of games that can be selectively executed.

FIG. 1




EP 1 113 407 A2

Printed by Jouve, 75001 PARIS (FR)

## Description

Background of the Invention

**[0001]** The present invention relates to a method of loading an application program into a smart card, smart cards, a method of loading scripts into a smart card, a terminal device capable of operating with smart cards, and a storage medium holding an application program; particularly to a computer system with highly-reliable security, especially, such system with its kernel built on an IC card wherein an application program stored into its nonvolatile storage can run inside the card.

**[0002]** IC cards (or termed smart cards) furnished with a built-in CPU (Central Processing Unit) that enables operations inside the card are expected to be used in various application fields, particularly, in financial application such as electronic money, and their introduction has been advanced positively in late years because of their information storage ability and highly-reliable security achievable.

**[0003]** Recently, operating systems (OSs) for such cards, enabling the safe coexistence of multiple applications on a single card have been generally used. As these OSs for such cards that support multiple applications on the card, "MULTOS" supplied by Mondex International and "Java Card" (TM) supplied by Sun Microsystems, Inc. are known. Smart cards with this kind of multi-application OS are controlled so that the application programs installed on the card will be highly independent of each other when running. Not only a plurality of programs can safely coexist on these cards, but also a new application can be added to these cards after a card is issued or an unnecessary application program can be removed from them. Thus, these cards can be regarded as safe computers rather than simple information storage media. From the viewpoint of active use of their highly-reliable security feature or new cards that supercede the conventional magnetic card function, smart cards are expected to have applications in the financial field such as credit cards or electronic money, especially, as the implementation of interlinking a plurality of applications.

**[0004]** Conventionally, a point system or a customer-loyalty system (hereinafter referred to as a point system) has been generally used as a means of getting more customers. This system is defined as "a system such that a customer's points increase by the use of the customer's card and the customer can be granted a predetermined service according to the accumulated points." Supposing that customers expect to be granted some privilege by getting points, shop managers and card issuers aim at the effect of promotion of using cards for shopping at their shops. Examples of such system are stamp cards that are valid only in a shopping district, department stores' point systems, or airlines' mileage programs. As one example, a department store's point system is explained below. Customer members have their cards issued from the department store. Whenever a member customer does shopping at the store and presents his or her card to the clerk, points the customer gets, according to how much he or she paid (for example, 20 points are added per 1,000 yens of payment) are accumulated and recorded in the customer's purchase log. When predetermined points have been accumulated, the customer can exchange the points for a gift certificate (for example, the customer can exchange 1,000 points for a gift certificate of 1,000 yens. In other words, member customers gain by a discount rate that is 1, 000 yens per shopping amounting to 50,000 yens, according to the calculation in this case.) Department stores may offer an additional discount in such a manner that the points are added at a double rate during a special campaign period or that if the amount a member customer paid for shopping or service at the department store a year reaches a certain amount, his or her discount rate rises. In this way, department stores usually stimulate the will to buy of their customers. For airlines' mileage programs as another example, flight distance of travel per customer instead of the amount the customer paid is accumulated. In the system of this kind, if the total distance that a customer traveled by using an airline reaches a predetermined flight distance, the airline grants the customer some privilege such as a free airline ticket or a seat upgrading service. In this case, similarly, the airline offers service in accordance with the log of a member customer who has used the airline, thereby motivating customers to select the same airline again. By installing such point system on a smart card, points of the card user can be correctly managed by means of the card. For a smart card with the multi-application OS, linking with electronic money or with credit card facilities can use the point system more effectively.

**[0005]** As one application that utilizes the feature of the above-described smart card supporting the compatibility of multiple applications, a "point system with game on smart cards" has been proposed. This system is such that game program is integrated with a point system on the card and point value may increase according to the result of the game stored in the card. The patents for the inventions regarding this system were applied for in Japanese Patent Application No. Hei 10-239812 and Japanese Patent Application No. Hei 10-321684. In these inventions, the count of user-playable games is defined as "rights to play game", and a method in which the smart card program can implement game application safely by managing the rights to play game and the points as the result of games has been proposed.

**[0006]** Moreover, another system in which a plurality of specific programs can be incorporated into a point management program has been proposed as a method of managing a point system on smart cards. The patent for the invention regarding this system was applied for in Japanese Patent Application No. Hei 10-307216. According to this method, by embedding shop-specific programs into the point management program, points from

a plurality of shops can be managed on a shop-by-shop basis by running a single program of point application.

Summary of the Invention

**[0007]** The multi-application smart card OSs such as "MULTOS" have a predetermined loading mechanism in view of security. The loading mechanism is used to check that the downloaded application is not falsified, that the authorized programmer has programmed the application, and that the card is granted the permission to download the application program. For example, checking to see whether the application program is falsified is performed as follows. As signature data, a hash value of the application program, encrypted in the secret-key crypt system of the Certificate Authority (CA) is attached to the application program. This hash value as the signature is compared with a hash value recalculated on the card for a match and thereby verification can be performed. Checking the above matters is important, on which the safety of the smart card is dependent. Thus, a strict procedure for each card type is prescribed and the mechanism is designed so that the application program transferred to the card cannot be downloaded unless it is coded in predetermined data format. This regulation is called an "application issue scheme."

**[0008]** Accordingly, in order to load an application program into a smart card with the multi-application OS is installed, a predetermined application authentication and registration procedure must be carried out, according to the above application issue scheme. Consequently, actual operation of replacing an application program installed on the card by another program requires considerable time and labor, though this is, in principle, possible after the card is issued. This is inevitable for maintaining the safety of the smart card. Notwithstanding, this problem is not considered significant for ordinary financial applications for which program contents do not change much frequently.

**[0009]** For game applications, however, frequent updating of their contents is required much, because users may tend to lose interest in playing a game unless varieties of games are available. Considering that the application authentication and registration procedure must be carried out each time a new game is loaded, while frequent game exchange is desirable, such complex procedure would deter us from turning game application features to full account.

**[0010]** Another problem arises with separately developing and distributing different application programs for different types of games. When points acquired by playing an old type game are transferred into a new type game, some procedure is required and point management in view of this transfer becomes complex.

**[0011]** In the application development phase, separately programming applications with similar facilities by each request is much time-consuming process. During developing and distributing many types of game applications, management of issues and management of distribution when applications are loaded into cards are required.

**[0012]** Furthermore, in the current situation where different OS types for smart cards such as "MULTOS" and "Java Card" coexist, an OS incompatibility problem further increases the reprogramming time. Game applications that run on smart cards under different OS types must be rebuilt separately on a plurality of platforms that use different OS types whenever game exchange occurs.

**[0013]** These problems are not limited to game applications, but similar problems are expected to occur with applications to run on smart cards for which frequent update of the contents for processing is desirable.

**[0014]** An object of the present invention is to provide a game application program to run on a smart card, making it possible to increase game variations to run in the program without being accompanied by a complex procedure of application program replacement, whereby the card user can readily play with one of a plurality of types of games on the card and new games can be developed independent of the difference between the OSs under which they are to run.

**[0015]** If the invention is extended to general applications beside game applications, another object of the present invention is to provide an application program to run on a smart card, making it possible to increase process variations to run in the program without being accompanied by a complex procedure of application program replacement, whereby the card user can readily request one of a plurality of types of processes on the card and new processes can be developed independent of the difference between the OSs under which they are to run.

**[0016]** In order to attain the above objects, means to run one of a plurality of types of games in a single application program installed on a smart card are proposed.

**[0017]** A conceivable way that is considered primary is sharing the entities (data storage and processor) for the point data as the result of playing games and the rights to play game with a plurality of games. Once the entities of managing the point data and the rights to play game have been set to commonly run in the game program processing, virtually, it can be thought that only the algorithm for judging game result differs depending on the game that is requested to run

**[0018]** Through close examination of types of games to primarily run on smart cards that are not regarded as having complicated calculation capability, it is obvious that processes required to execute games are "receiving data sent from the user via the terminal," "generating a random number," "simple addition/subtraction," "storing data," and "data comparison and branching" in combination that are iterated. If part of an application program is made modular, that is, it is made up of "components" that independently implement the above proc-

esses, games can be defined in "scripts" like representation that defines sequence in which these components are called. Specifically, preparing processing modules, namely "components" to implement the processes required to execute games and an "interpreter" for interpreting and executing scripts in a single application program is essential. This makes it possible to run one of different games by selectively executing the game definition "scripts" generated outside the program.

**[0019]** If such scripts are permitted to be loaded from outside or unloaded if necessary as part of an application program through a terminal, it becomes possible for a single game application program on a smart card to offer a plurality of games of different types that can be selectively executed.

**[0020]** Exchanging or adding scripts, if assumed feasible, however, means that any script can be stored into an application program and there is a possibility of including a game in ill-intentioned scripts, which may result in that some user could get points by foul play with such game. The security of smart cards is satisfactory, but becomes useless for such foul play. As the substitution for the application program issue scheme defined as the card OS security mechanism, a pseudo issue scheme must be provided within an application program installed on the card to control storing scripts and prevent ill-intentioned scripts from being stored. Specifically, a controller must be provided to control loading and unloading scripts so that ill-intentioned scripts will be shut out from the application.

**[0021]** The present invention assures safety of loading scripts by providing the application program installed on the smart card with the pseudo issue scheme instead of the application program issue scheme that the OS of the card has. The invention also prepares the processing entity for interpreting and executing scripts within the application program, so that a single application program can offer types of games which are different features, though limited.

**[0022]** Points may increase, according to the result of playing with a game, and the player can later be granted a predetermined service (for example, exchanging points for a commodity), according to the points. Data of these points, of course, can be processed commonly for a plurality of games and in addition can be managed for each game issuer by adding game issuer information to the game definition scripts stored into the card.

**[0023]** Therefore, as the means to solve the above-mentioned problems, the present invention comprises the following six major entities.

**[0024]** As the means to be provided on the smart card side.

(1) An application program consisting of the following elements:

(a) Game executing components: A set of modules for implementing the processes programmed in the card, required to execute the game application;

(b) Storage for game definition scripts: Area for storing scripts that define sequence in which the components are to be executed;

(c) Script interpreter: Interpreter for interpreting and executing game definition scripts;

(d) Storage and processor for point data: Area for storing points that may increase, according to the result of playing a game, and the processor for point data management;

(e) Storage and processor for rights to play game: Area for storing the count of rights to play game and the processor for managing the rights to play game; and

(f) Command analyzer: Processor to analyze commands from a terminal and call the appropriate process within the program.

The above are necessary. In addition,

(2) Processor for storing game definition scripts: The processor has the function that manages storing game definition scripts and exchanging scripts. Processing of this processor is based on the game issue scheme prescribed in the application program.

(3) Function of point management per issuer: Manages points and rights to play game per issuer.

The above two functional entities are prepared as required.

The following are required for a terminal device for operating with the smart card in question:

(4) Function of issuing a game: Issues game definition scripts and/or rights to play game to the smart card by performing data communication with the smart card under a predetermined protocol. Game definition scripts and rights to play game may be separately managed or put under integrated management.

(5) Function of executing a game: Executes a game by sending user-input commands for playing game to the card and receiving responses from the card by performing data communication with the smart card under a predetermined protocol. A user interface that allows the user to play games is also required.

(6) Function of calculating points: Obtains the point count stored in the card and sets a new point count (by subtraction, primarily) by performing data communication with the smart card under a predetermined protocol. Point calculation is executed with an issuer identifier if point data management per issuer is performed.

**[0025]** A single terminal may be provided with all of the above functions in items (4) to (6) or separate terminals may take part of the functions.

**[0026]** Other and further objects, features and advan-

tages of the invention will appear more fully from the following description.

Brief Description of the Drawings

**[0027]** A preferred form of the present invention is illustrated in the accompanying drawings in which:

Fig. 1 is a conceptual diagram illustrating how a game and point system using smart cards serves customers;
Fig. 2 is a diagram illustrating the information flow during program execution on a smart card that supports the compatibility of multiple applications on the card;
Fig. 3 is a diagram for illustrating a concrete smart card configuration for the game and point system concerned with a preferred embodiment of the present invention (the first one);
Fig. 4 is a diagram for illustrating a concrete smart card configuration for the game and point system concerned with a preferred embodiment of the present invention (the second one);
Fig. 5 is a diagram for illustrating a concrete smart card configuration for the game and point system concerned with a preferred embodiment of the present invention (the third one);
Fig. 6 is a diagram for illustrating a concrete smart card configuration for the game and point system concerned with a preferred embodiment of the present invention (the fourth one);
Fig. 7 is a diagram for illustrating a concrete smart card configuration for the game and point system concerned with a preferred embodiment of the present invention (the fifth one);
Fig. 8 is a diagram for illustrating a concrete smart card configuration for the game and point system concerned with a preferred embodiment of the present invention (the sixth one);
Fig. 9 is a diagram illustrating an example of game issue operation (the first one);
Fig. 10 is a diagram illustrating an example of game issue operation (the second one);
Fig. 11 is a diagram illustrating the processing of the script interpreter;
Fig. 12 gives an example of scripts described in a language to call common process components;
Fig. 13 illustrates a slot machine game example;
Fig. 14 gives an example of scripts described for the slot machine game;
Fig. 15 illustrates a roulette game example;
Fig. 16 gives an example of scripts described for the roulette game;
Fig. 17 illustrates a shooting game example;
Fig. 18 illustrates an example of the processing procedure of the shooting game;
Fig. 19 gives an example of scripts described for the shooting game; and
Fig. 20 is a diagram illustrating an example of questionnaire system operation.

Detailed Description of the Preferred Embodiments

**[0028]** Fig. 1 shows a conceptual diagram illustrating how a game and point system using smart cards serves customers. The system is roughly divided into the service providers' side (101) and the users' side (109). The service providers' side (101) comprises an application provider (102) that distributes a game application by loading it into smart cards and terminals, a service administrator (103) that administrates game operation, and shops (104) at which a game issue is loaded into a smart card of a user and points are exchanged for a service. Here, a single entity may double the application provider (102) and service administrator (103) or the application provider (102) and service administrator (103) may be separate entities. In normal situation, a plurality of shops (104) exist in separate sites. In some case, a shop (104) exists only in one site and belongs to the same application provider (102) and service administrator (103). The users' side (109) consists of one or more users (113) and each user (113) possesses at least one smart card (110). The user (113) may possess a game playable terminal for user (111) provided with the input/output function for executing a game application installed on the card (but, this terminal is not necessary). The game playable terminal for user (111) need not be limited to that only having facilities dedicated to playing games, but also may be an ordinary personal computer (PC) with a smart card reader/writer (the smart card accessing device) or a special portable terminal for operating with smart cards, having facilities such as checking the user's balance recorded in the card, provided that it has at least the function of data input /output to/from a smart card (110) and a support (as an auxiliary) of game application execution.

**[0029]** The application provider (102) first loads a game application into the smart card (110) of each user (113). In the typical case, a game application is loaded into the card when the smart card (110) is issued to the user (113). After the issue of the card, a game application can be freely loaded into the card, which is a feature of the smart card on which the OS supporting the compatibility of multiple applications on the card is installed. The information on loading applications may be reflected in game management data (106) which will be described later.

**[0030]** The service administrator (103) retrieves necessary data from the game management data (106) database in which all kinds of information on the game and point system in question are filed and uses a game management server (107) to distribute game parameters to terminals (108) at shop.

**[0031]** Shops (104) issue rights to play game or game patterns to smart cards (110), according to how much the user (113) has paid (shopped) at the shop. The rights

to play game mean the rights for the user to play with a game and the rights decrement by one after the user plays one game. The game patterns mean types of parameters of games to be executed within the smart card, including rights to play game.

**[0032]** Having rights to play game given to the smart card (110), users (113) can play games installed in the card by means of the game playable terminal for user (111) or a game playable terminal at shop (112) placed in a shop (104) (As with the game playable terminal for user, the terminal managed by a shop may be in any form, provided it has the interface with a smart card and the game execution support function) . According to the result of game play, the points in the smart card (110) are updated.

**[0033]** The points can be exchanged for a commodity or a prize at a shop (104). Game issue data, user game play log data, and point exchange log data, whenever generated, are saved-and stored into the database of game management data (106), and fed back to the next process of generating game parameters and issuing a game. In this way, the service administrator dynamically controls game parameters by using the latest game data, so that the service provider (101) can manage overall system circumstances.

**[0034]** Now, information flow during program execution on a smart card that supports the compatibility of multiple applications on the card will be briefly explained with reference to Fig. 2. The smart card (201) is equipped with an I/O interface (202), and one or more applications (203) are assumed to have been loaded into it, according to a predetermined procedure. The application program (203) and the application program (203'), shown, are separate independent programs and each program is inhibited from making an illegal access to the other program. A terminal (120) is equipped with a smart card reader/writer (161) and an I/O interface (164), and a terminal OS (163) is installed on it, and moreover, one or more terminal programs (162) for processing with a mating application program installed on the smart card are installed on it. One or more terminal programs for processing with one application program on the smart card are normally required.

**[0035]** When the user (or a clerk) performs input (151) to the terminal (120) via a user interface, the terminal program (162) generates a command (152) to be sent to the smart card. The terminal program (162) sends the command to the smart card (110) (153) via the smart card reader/writer (161). When the smart card (110) receives the command, the smart card OS (201) determines an application program (203) to which the command is to be sent and sends the command (154) to the application program (203) that the terminal program (162) intends to send it to. The processing part (205) of the application program (203) executes processing in accordance with the command; it accesses the application data (204) database, performs value update (155), and generates a response (156). The response is returned (158) to the terminal (120), and the terminal program (162) shows results (168). This is program execution flow in a series.

**[0036]** Even if a plurality of application programs (203) have been loaded, the intended program execution on the smart card (110) can be performed via the terminal (120) by terminal program (162) switching to the appropriate application program (203) to which the command is issued. Basically, the application programs (203) on the smart card (110) are controlled not to affect the data for another program illegally. However, for example, under "MULTOS," a plurality of application programs can operate in linkage with each other by a predetermined method termed delegation (meaning that one program entrusts or transfers a process to another program). (The latest version of "MULTOS" is provided with this function. The delegation function is not described in detail herein.)

**[0037]** Next, concrete smart card configurations for the game and point system embodied in a preferred embodiment of the present invention will be explained with reference to Figs. 3 to 8. The following explanation will be made by using the case where two types of games are available on the card as an example. Here, the "types" of games are game variations for which different algorithms are used to define how game results are reflected in points, where algorithms shall be basically defined by the combination of the processes within the card, such as data calculation and random number generation.

**[0038]** Fig. 3 shows a smart card configuration that can be embodied to run game applications according to the previous invention for which the application for patent was filed (as Japanese Patent Application No. Hei 10-239812 and Japanese Patent Application No. Hei 10-321684), wherein two types of games are available on the card. The smart card (110) operates under the card OS (201) supporting the compatibility of multiple applications on the card, such as "MULTOS" and "Java Card" and is equipped with the input/output interface (202) for data transfer to/from the terminal (120). The specifications of the above previous invention state that one algorithm for result judgment is defined for one application each. Thus, as many game application programs (200) as the number of game types available on the card are necessary. In this configuration shown, a first game application is marked (A) and a second game application is marked (B).

**[0039]** The game application program (A) (200) comprises the data storage portions for game parameters (221), rights to play game (222), and point data (222) and actual processing parts, command analyzer (211), algorithm for result judgment (212), processor for rights to play (213), and processor for point data (214). The shaded portions in the figure, (211), (212), (213), and (214) are the processing parts that become fixed once the program has been loaded into the card. Data contained in the storage portions (221), (222), and (223)

changes with the program execution. Commands from the terminal (120) are roughly divided into those regarding game issue from a shop and those regarding game play from the user. The command analyzer (211) distributes commands to the processors, according to the command type, and the processors execute processing and thereby the points may increase and the rights to play game decrement. When receiving commands for playing a game from the user, the game is executed, according to the algorithm of result judgment (212) and game parameters (221) depending on the user input. According to the game result, the points in the point data (223) storage may increase and the count of the rights to play game (222) decrement by one a game. The count of the rights to play is controlled so that the user cannot play with a game in excess of the user-playable game count corresponding to the rights given to the user from a shop.

**[0040]** The game application program (B) (200') is also configured the same as the program (A) and include independent processing program components installed. Because commands from the terminal (120) are sent after application selection, independently set commands are issued to each of the game application programs (200) and (200'), thereby running the application program. Of course, point data (223) and (223') are also stored separately, and in principle, integrated processing for point exchange is impossible. Definitely different parts between game (A) and game (B) are the algorithms for result judgment (212) and (212'). Other components for processing of rights to play game and points consist of similar functional units. Notwithstanding, separate programs execute separate processing and this should be considered a problem in view of quiet an ineffective use of limited resources of smart cards.

**[0041]** To run the game application (A) program (200) and the game application (B) program (200') which are essentially separate applications under "MULTOS," it is necessary to carry out the application registration and authentication procedure for each program in accordance with the application issue scheme defined for the OS when installing the applications. An outstanding feature of smart cards supporting the compatibility of multiple applications on the card is that dynamic application loading and unloading are possible after the card issue. For applications such as game software for which frequent updating of the contents is desirable, however, whenever a new game program programmed to run with another algorithm is loaded, carrying out the application registration and authentication procedure is troublesome and burdensome. Moreover, when switching from game (A) to game (B) occurs, the user might wish the accumulated points as the result of the previous plays with the game (A) to be inherited for natural feelings. For the smart card configuration shown in Fig. 3, however, it is not simple to transfer points from one game to another and perform accompanied point information management.

**[0042]** As concerns counting points in common for a plurality of games, a possible method of resolving the program is using a special application for point management besides game applications. This method is feasible by the above previous invention. This method is such that a point application program in addition to game applications is prepared and point data management is performed within it. When requested from the game applications, point change processing is executed by using the delegation of the card OS (201). Even if the integrated point management problem is solved by using this method, however, the application registration and authentication procedure for a new game program to be added is still required and the above method does not provide a drastic solution.

**[0043]** Then, a method of processing a plurality of types of games within a single application will be considered below.

**[0044]** Fig. 4 shows a smart card configuration according to the method in which an game application is installed comprising common parts for point data for which integrated management is desirable and command processing and portions for specific processing such as game parameters and result judgement algorithm. The latter portions are prepared as many as the number of game types. On the smart card (110), a game application program (200) is installed, wherein two types of games, game (A) and game (B) can be processed. Although another game type, game (C), of course, can be incorporated into the program, the following explanation will discuss the example case where the application program accommodating two games is installed as shown. Note that the game types to be executed are fixed when the application is issued and adding a new game cannot be performed in this example.

**[0045]** Point data (223) storage is common so that points acquired as the result of game play can be processed common for a plurality types of games. The command analyzer (211) that receives commands from the terminal and supplies them to the appropriate processor is also common to a plurality of games, separate from the game processing components. Game parameters (221), rights to play game (222), and algorithm for result judgment (212) are provided as many as the number of game types as the components to execute different processing for different games. The processor for rights to play (213) and processor for point data (214) are provided as common processors.

**[0046]** Establish a command identification scheme in advance so that commands sent from the terminal (120) can be identified, according to the game type. The command analyzer (211) analyzes commands it receives, finds which game type to which the commands are supplied, and selects the appropriate algorithm for the commands. If the commands are those for game (a), processing is executed, according to the algorithm for result judgment (212) for game (A) and the point data (223) may be updated, according to the result of the

processing. Updating the points in the point data (223) storage is performed in the same way whether game A or game (B) has been executed.

**[0047]** Fig. 5 shows a smart card configured according to a game execution method upgraded from the corresponding method for the card shown in Fig. 4. Basically, most of game processes can be represented as the iteration of processing user input values, generating a random number in the card in accordance with the input timing, and judging the result from the combinations of local variables and constants given as parameters. Even if the algorithm for result judgment differs between games, most of other portions of a game application can be executed in common. Thus, part of the game application is made modular for processes that can be executed in common for a plurality of games, such as generating a random number, addition/subtraction, and iteration as common process components. Common process components (244) are prepared as component modules for executing processes that are required in similar context.

**[0048]** With the focus on game (A), the application includes the storage areas for game parameters (221) and rights to play game (222) and definition of component call (225) in the game (A) main (215) defines the procedure for judging whether the user wins at the game, that is, what sequence in which the common process components (224) are to be called. The program is executed in almost the same way as for the method used for the smart card shown in Fig. 4. The command analyzer (211) selects the appropriate game main, according to the command type. Thereby, a plurality of games, as assembled into the program beforehand, can be processed. For the smart card example shown in Fig. 5, as compared with that shown in Fig. 4, the processes that can be executed in common are packaged as components in as large part of the program as possible, so that efficient application operation can be achieved.

**[0049]** Whether in the game execution method for the card shown in Fig. 4 or in the corresponding method for the card shown in Fig. 5, a plurality of game types assembled in advance into the application as the application was programmed can be processed and one of different games can be executed, the appropriate one being selected in accordance with the commands from the terminal, but another game type cannot be assembled into the application once the programmed application has been issued. To enable another game type to be executed on the card, it is necessary to program a new application that accommodates the game and replace the existing application on the card by the new application, when the application registration and authentication procedure must be carried out again.

**[0050]** In Fig. 6 and Fig. 7 that follow, a method is proposed in which replacing a game type by another game type or adding another game type can be performed without being accompanied by the application registration and authentication procedure and application replacement.

**[0051]** Specifically, game definition scripts supercede the definition of component call (225) used in the game execution method for the card shown in Fig. 5 and a script interpreter for interpreting and executing the game definition scripts is prepared in the processing part of the game application program. Separate suits of the scripts that are replaceable define different games and the interpreter practically interprets and executes each suit of the scripts.

**[0052]** Fig. 6 shows a smart card configuration with a game application being installed on the card, the application programmed according to the method in which game definition scripts are described and executed by the interpreter. Here, again, the shaded portions are fixed components of the program that are predetermined.

**[0053]** In this example, game definitions and rights to play game are handled as separate entities. In the same,manner as for the examples shown in Fig. 1 to Fig. 5, the rights to play game (222) and the point data (223) are processed by the processor for rights to play (213) and the processor for point data (214) in common for all types of games. The common process components (224) are predefined and the game definition scripts (226) define the algorithm for result judgement that differs for each game by describing sequence in which these components are to be called.

**[0054]** These game definition scripts (226) are replaceable and a processor for exchanging game scripts (216) controls loading/unloading the scripts. Despite that the strict application issue scheme is prescribed to assure the smart card security, it cannot be denied that there is a possibility of safety loss due to making game exchange in units of game definition scripts possible for simple and convenient game exchange. Within the application program, therefore, the following must be checked: the game definition scripts to be stored is not falsified; the authorized programmer has programmed the scripts; and the game application is granted the permission to store the scripts. Specifically, a particular game issue scheme instead of the OS's application issue scheme must be prepared within the application and storing game scripts must be controlled so that safe game scripts only will be installed on the card, based on this scheme. The processor for exchanging game scripts (216) fills this controlling role.

**[0055]** The rights to play game (222) are data that defines how many times the card user can play with what type of game and this data is stored into the card when the game is issued. When the card receives the commands for playing a game from the terminal, if the rights to play game (222) exist, the script interpreter (215) interprets and executes the contents of the game definition scripts (226) for the game. When the game play finishes, the rights to play game (223) decrement by one as explained for the fundamental card configuration.

The card configured as shown in Fig. 6 can accommodate a plurality of game types by storing game definition scripts (226) for each game type into the game application on the card. In addition, under the control of the processor for exchanging game scripts (216), a game type can be replaced by another game type even after the application is loaded into the card.

**[0056]** For the smart card example shown in Fig. 7, which resembles the example shown in Fig. 6, the data on the rights to game play is included in game definition scripts. That is, the game definition scripts (226) are programmed to include one right to play game (may include a plurality of rights to play game, not limited to one right) . Once a game has been executed once (or two or more times corresponding to a predetermined playable game count) described in the game definition scripts, control means is required to delete the game scripts so that the same scripts will never be executed again thereafter. The rights to play game and the game definition scripts are managed together under a single processor, namely a processor for storing game data (217). This processor also deletes game scripts that have been executed completely as well as stores game scripts sent from the terminal when the game is issued into the data area. Needless to say, the processor for storing game data (217) must perform the script safety check as the processor for exchanging game scripts (216) shown in Fig. 6 does.

**[0057]** Fig. 8 shows a game application configured to adapt the game execution method for the card shown in Fig. 6 for a plurality of issuers of points. The rights to play game (222) are stored, each data thereof comprising a triple of entries: issuer, game ID, and play count. Accordingly, points are stored per issuer in the storage area for point data (223). When the card receives the commands for playing a game sent from the terminal, the rights to play the game (222) are checked. For the game type for which the rights to play remains, the script interpreter (215) executes the game, according to the game definition scripts (226). Following the game execution, points may be added to the point data (223) for the issuer of the game and the rights to play game (222) for the issuer of the game that has just been executed decrement by one.

**[0058]** This method is application of the point system concept according to the above-mentioned pervious invention for which the application for patent was filed (as Japanese Patent Application No. Hei 10-307216), wherein the point data (223) is retained separately for a plurality of issuers, but not unified. In this method, the card can accommodate not only a plurality of game types, but also a plurality of point service providers. Using this method, game service administrators can offer shops a game application rental service which would expand the potential availability of game applications.

**[0059]** With reference to Fig. 9 and Fig. 10, now, system operation examples of the game and point system using either of the smart cards configured as illustrated in Fig. 6 and Fig. 7 will be explained below. In the system operation example illustrated in Fig. 9 wherein the smart card shown in Fig. 6 is used, game definition scripts and rights to play game are separate and game exchange is assumed to be performed at suitable timing. In the system operation example illustrated in Fig. 10 wherein the smart card shown in Fig. 7 is used, game definition scripts and rights to play game are put together and the scripts of a game are invalidated (deleted) whenever the game has been executed completely.

**[0060]** Fig. 9 is a diagram illustrating an example of the game and point system operation using the smart card on which a game application accommodating a plurality of games is installed. In this example, game definition scripts and rights to play game are separate.

**[0061]** A game application program (200) is assumed to have been loaded into the smart card (110). The game application (200) comprises a command analyzer (211), a script interpreter (215), a processor for exchanging game scripts (216), a processor for rights to play (213), a processor for point data (214), and other components. Point data (223) is stored as points managed per issuer Company. Rights to play game (222) are also managed per issuer and the game type and playable count per issuer are stored. Game definition scripts (226) are game contents defined by describing sequence in which the common process components (224) are to be called. In fact, the script interpreter (215) interprets the contents of the scripts and executes the scripts. The game definition scripts (226) can be exchanged for new scripts as required.

**[0062]** The processing procedure for each phase of game exchange, game issue, game play, and exchanging points for a service will be explained below.

(Game exchange)

**[0063]** From a game maintenance terminal (114) at a shop (104) or any other place, a command to exchange game scripts (132) and game definition data (scripts) (141) are sent to the smart card (110). Then, once the command analyzer (211) has interpreted the received command as the command to exchange game scripts (132), the processor for exchanging game scripts (216) executes the processing of game definition exchange. At this time, it must be assured that the game definition scripts received from the terminal have been issued from the authorized issuer because the scripts can rewrite the point data. For the assurance thereof, all game definition scripts (141) issued must include a cipher key that authenticates the authorized issuer of the scripts, the key encrypted in accordance with the game issue scheme predefined within the application. According to the cipher key, the processor for exchanging game scripts (216) must perform authentication of the scripts it received. Furthermore, program exchange may be necessary on the terminal side when a new game is issued, because a terminal program appropriate for the

new game is required on the game playable terminal (111) when the user plays the game.

(Game issue)

**[0064]** According to how much the user paid for shopping or service at a shop (104) of Company (X), a terminal at shop (108) sends the smart card (110) a command to add the right to play game (133) and right to play data (142). The right to play data (142) contains information that the user can play with the game (A) once and the issuer is Company (X). To assure safety, encryption is desirable for this data as well. The command analyzer (211) interprets the command and the processor for rights to play (213) adds the right to the rights to play game data (222).

(Game play)

**[0065]** When the user (113) is playing with a game, commands for playing game (131) are sent to the smart card (110). If the rights to play game (222) exist, the user can play the game of the appropriate type. Following the game play, the points retained in the point data (223) storage may increase, wherein the game result may update the points for the issuer of the rights to play game included in the rights to play game data (222) for the game just executed. After the game play, the rights to play game (222) decrement by one. To enable game execution, a game executing terminal program appropriate for the game type within the smart card must exist in the game playable terminal (111).

(Exchanging points for a service)

**[0066]** Points accumulated as point data (223) while the card user has so far played games can be exchanged for a predetermined service at a shop (104). When the card user wants to exchange points for a service, a command to calculate points (134) is sent to the smart card (110) from a terminal at shop (108) (that may be the same as the terminal from which rights to play game are issued or different from such terminal). The command analyzer (214) passes the processing control to the processor for point data (214) and the processor for point data (214) executes subtraction of points. If the shop (104) from where the command was sent belongs to Company (X), only the points issued by the Company (X) can be exchanged for a service.

**[0067]** In the system operation scheme illustrated in Fig. 9, the user can play games of as many types as the number of the suits of game definition scripts (226) (of course, only if the rights to play game exist in the card). By replacing game definition scripts (226) with new scripts as required, the game application (200) can continue to offer a new game without being replaced.

**[0068]** Fig. 10 is a diagram illustrating another example of the game and point system operation using the smart card on which a game application accommodating a plurality of games is installed. In this example, game definition scripts and rights to play game are unified. As is the case with the system operation example in Fig. 9, it is assumed that a game application program (200) have been loaded into the smart card (110). The game application (200) comprises a command analyzer (211), a script interpreter (215), a processor for storing game data (217), a processor for point data (214), and other components. Point data (223) is stored as points managed per issuer Company. Difference from the example in Fig. 9 is that game data (227) is stored, each data comprising a pair of game definition scripts and issuer information. Similarly, the script interpreter (215) executes game definition scripts by interpreting the contents of the scripts. A suit of scripts can be regarded as one right of the user to play game. Game execution may rewrite only the point data (223) for the issuer of the game. Once the game has finished, the suit of the game scripts is invalidated by deleting it so as not to be executed again.

**[0069]** For this example, the processing procedure for each phase of game issue, game play, and exchanging points for a service will be explained below.

(Game issue)

**[0070]** According to how much the user paid for shopping or service at a shop (104) of Company (X), a terminal at shop (108) sends the smart card (110) a command to store game scripts issued (135) and game data (143) containing game definition scripts and issuer. To assure safety, a cipher key that authenticates the authorized issuer of the scripts, encrypted in accordance with the predefined scheme, must be attached to the game data (143). According to the cipher key, the processor for storing game data (217) must perform authentication of the scripts it received. Furthermore, program exchange may be necessary on the terminal side when a new game is issued, because a terminal program appropriate for the new game is required on the game playable terminal (111) when the user plays the game.

(Game play)

**[0071]** When the user (113) is playing with a game, commands for playing game (131) are sent to the smart card (110). At this time, if the game data (227) exists, the user can play the game described in the game definition scripts. Following the game play, the points retained in the point data (223) storage may increase, wherein the game result may increase the points for the issuer included in the game data (222) for the game just executed. After the game play, the game data (227), the scripts of the game are erased so as not to be executed again. To enable game execution, a game executing terminal program appropriate for the game type within the smart card must exist in the game playable terminal

(111).

(Exchanging points for a service)

**[0072]** Points accumulated as point data (223) while the card user has so far played games can be exchanged for a predetermined service at a shop (104). When the card user wants to exchange points for a service, a command to calculate points (134) is sent to the smart card (110) from a terminal at shop (108) (that may be the same as the terminal from which rights to play game are issued or different from such terminal). The command analyzer (214) passes the processing control to the processor for point data (214) and the processor for point data (214) executes subtraction of points. If the shop (104) from where the command was sent belongs to Company (X), only the points issued by the Company (X) can be exchanged for a service.

**[0073]** In the system operation scheme illustrated in Fig. 10, because every game data (227) includes game definition scripts, there may exist duplicated suits of scripts in the card if a plurality of same type games are stored, which may be considered inefficient. However, the flexibility of game type selection increases as compared with the example illustrated in Fig. 9 wherein game definition scripts are replaced by new scripts as required. Because issuing the right to play game means issuing a new game, the game application (200) can continue to offer a new game without being replaced. Whenever a new type game is developed, however, the associated program for playing the game on the terminal needs change. In a situation where game playable terminals are distributed, updating the programs on the terminals is likely to be time-consuming work.

**[0074]** Next, the flow of processing of the script interpreter will be explained with reference to Fig. 11. In the game definition data (300) area, parameters (301), rights to play (302), and scripts description (303) are stored. Separate from the game data, point data (308) exists.

**[0075]** The script interpreter has a program counter (304) as a local variable and work arrays (305) and is able to read a command parameter (306) sent from the terminal and rewrite a response parameter (307) to be returned to the terminal.

**[0076]** When a command is sent to the smart card from the terminal, the script interpreter receives and analyzes the command from the terminal (311), and selects the specified game type (i.e., it calls the specified game definition scripts) (312). At this time, the interpreter stores a parameter included in the received command as the command parameter (306).

**[0077]** The interpreter resets (313) the program counter (304) and begins to execute the game scripts. The interpreter executes in order the scripts described in the scripts description (303) part; it basically increments the program counter one by one (314) while analyzes the scripts in order (315) and calls appropriate common process components, thereby executing the scripts. If the interpreter encounters a "wait for command" script, it immediately returns the response to the terminal and awaits the next command reception (316). If the interpreter encounters an "end" script, it returns the response to the terminal and exits from the script execution process (317). If the interpreter encounters a script in which a jump or a branch is specified, it resets the program counter to the jump address and proceeds to analyzing the next script (318). If the interpreter encounters any other script, it calls the appropriate common process component associated with the script and actually executes the script (319), and returns to the step of incrementing the program counter by one (314). During the script execution (319), the interpreter reads a value of command parameter (306) while updates a value in the work arrays (319) and a value of response parameter (308). By continuing this processing, the game application is run.

**[0078]** Fig. 12 illustrates an example of scripts described in a language to call common process components. Here, array [ ] is one of the work arrays (305), cmd [ ] denotes a command parameter (306), and rsp [ ] denotes a response parameter (307). Each script is to call one common process component and fills the role of running the application program. Up to three parameters are assigned to one script.

**[0079]** In this example, scripts are described in a likely usual programming language to facilitate understanding. Considering that the scripts are executed on the smart card with small storage capacity, a more specialized language may be suitable for describing the scripts limited to game application (for example, representation in byte strings, each byte consisting of fewer bits) . A method may be used in which any compiling means is prepared on the terminal side from which scripts are issued to translate scripts into those represented in byte strings and the translated scripts are issued.

**[0080]** With reference to Figs. 13 through 19, examples of three types of games that the program application can offer will be given below, wherein game definitions are represented by using the scripts defined in Fig. 12.

**[0081]** Fig. 13 shows examples of slot machine game play screens appearing on the game playable terminal, wherein three boxes (corresponds to three random numbers) are shown and a range of settable values for a random number represented in one box is 0 to 2. The values settable in each box correspond to three fruit symbols: "1" for an apple symbol, "2" for a Japanese orange symbol, and "3" for a banana symbol.

**[0082]** The game begins with screen (a) . The current points (401) and remaining games (402) that the user can play are shown and values in three boxes (405) are not fixed yet with the reels rotating on the screen as depicted. The user pushes three "Push" buttons (406) one by one. By each button push, a command to generate a random number is sent to the card and the symbol

corresponding to a fixed value of the random number is displayed in the box.

**[0083]** Screen (b) is an example screen after the first user choice by the "Push" button, A random number of "0" is assumed generated in the card and the apple symbol is displayed in the corresponding box. This is repeated three times.

**[0084]** Screen (c) is an example screen after the third user choice by the "Push" button, when one game is over (the user wins). Because of matching of the symbols in the three boxes, a message (403) appears, indicating that the game result is "you win" and "100" points the user gained at the game are added to the points. The points increase accordingly.

**[0085]** Screen (d) is another example screen after the third user choice by the "Push" button, when one game is over (the user loses) . Because of mismatching of the symbols in the three boxes, a message (404) appears, indicating that "you lose" and no points are added.

**[0086]** Fig. 14 shows the scripts description example for the slot machine game shown in Fig. 13, described by using the scripts in Fig. 12. The game is represented in 18 lines of scripts, lines 00 to 11 (binary).

Lines 00 to 02: To be called upon the first "Push" button push. One random number is generated and stored into a work array and the fixed value of the random number is returned. To return the operation result to the terminal, a "Return" script is defined.

Lines 03 to 05: To be called upon the second "Push" button push. One random number is generated and stored into a work array and the fixed value of the random number is returned. Lines 06 to 08: To be called upon the third "Push" button push. One random number is generated and stored into a work array. A value to indicate "the last" is placed in the response and the processing proceeds to result judgment. Lines 09 to 11: Result judgment. If the stored values of three random numbers all match, predetermined points are added to the current points. Finally, the result is returned to the terminal and the processing terminates.

**[0087]** Fig. 15 shows examples of roulette game play screens appearing on the game playable terminal, wherein the win probability is one third and the user is given three betting chances. Three possible values of random numbers correspond to "A," "B," and "C." Illustration in color might be easier to see, though not be presented here.

**[0088]** Screen (a) is the initial screen. The current points (401) and remaining games (402) that the user can play are shown. To begin the game, choose one of betting places (407) A," "B," and "C" before the roulette (408) rotates.

**[0089]** Screen (b) is a screen that appears, following the game start. Here, the user is assumed to have chosen "B" out of the betting places (407) and a bullet is placed in the B box. Score (410) and remaining chances (411) are shown. Because the user is given three betting chances for this game, "3" is shown as the remaining chances (411). When the roulette starts to run as the user pushes the "Start" button (409), the remaining chances (411) decrement by one.

**[0090]** Screen (c) is a screen wherein the roulette (408) is rotating (this state appears on the screen, while the program on the card side waits for command input) . When the user pushes the "Stop" button (to stop the roulette running) (406), the associated command is issued to the smart card.

**[0091]** Screen (d) is a screen when one roulette run is over. According to the value of the random number generated in the card, that is, if there is a match between the bet place you chose and the value, a "You Win" (403) message appears and the score increases.

**[0092]** Screen (e) is a screen when one roulette run is over and you lose. A "You lose" (404) message appears and the score does not increase.

**[0093]** One game ends with screen (f) after three roulette runs with a random number generated. A "GAME OVER" message and the score count are displayed (412).

**[0094]** Fig. 16 shows the scripts description example for the roulette game shown in Fig. 15, described by using the scripts in Fig. 12. The game is represented in 27 lines of scripts, lines 00 to la (binary).

Lines 00 to 05: To be called upon the first "Stop" button push. The user-input value received as a command parameter is stored into a work array and compared with a random number generated and the result of comparison is stored into another array. The comparison result and the value of the random number are returned to the terminal.

Lines 06 to 0b: To be called upon the second "Stop" button push. The user-input value received as a command parameter is stored into a work array and compared with a random number generated and the result of comparison is stored into another array. The comparison result and the value of the random number are returned to the terminal. Lines 0c to 10: To be called upon the third "Stop" button push. The user-input value received as a command parameter is stored into a work array and compared with a random number generated and the result of comparison is stored into another array.

Lines 11 to la: Result judgment. According to the three comparison results, points are added to the current points. Finally, the result is returned to the terminal and the processing terminates.

**[0095]** Fig. 17 shows examples of shooting game play screens, wherein five boxes appear and the user is given five chances to play. The user clicks at one of the boxes to hit the target that moves at random, appearing in any box a second. If the target appears in the box at which the user clicked, the user wins. Clicking can be repeated five times.

**[0096]** Screen (a) is the initial screen. The current points (401) and remaining games (402) that the user can play are shown. To begin the game, click the "Start" button (409).

**[0097]** Screen (b) is a screen before the user clicks at a box on the screen. In any of the five boxes (405), the target symbol (413) appears at random. The target position changes, according to the value of the random number that is generated per second. In the screen, the current game score (410) and remaining chances (411) to play are displayed.

**[0098]** Screen (c) is a screen when the user clicks at a box on the screen. In this case, because the target appears in the box that the user has chosen, a "click - hit" (414) message appears and 10 points are added to the score.

**[0099]** Screen (d) is another screen when the user clicks at a box on the screen. If the box that the user has chosen differs from the box where the target appears, a "click - lose" (415) message appears.

**[0100]** The game ends with screen (e). When the user finishes using five play chances, a "GAME OVER" message and the score count are displayed (412). If the count of the remaining games (402) is other than 0, a new "Start" button (409) appears.

**[0101]** Fig. 18 illustrates the procedure of processing between the terminal (120) side and the card (110) side regarding the shooting game shown in Fig. 17. In this procedure that is the same for other programs on the smart card, basically, the terminal (120) sends the card a command and a parameter and the card (110) executes processing in accordance with the received command and returns a response to the card. Because the smart card on which the current version of MULTOS and other OSs is installed does not have a built-in clock (the operation on the card side is triggered by command reception from the terminal), a request for generating a random number must be sent per second from the terminal in order that the card generates a random number a second to update the screen.

**[0102]** First, the terminal (120) sends a command (parameter = 0) (321) that requests the card to generate a random number.

**[0103]** The card (110) receives the command (331), generates a random number in the value range corresponding to the number of boxes, assigns the random number to variable R (332), and sends a response with parameter = R back to the terminal (333). The terminal receives the response and updates the screen (322), according to the value of R. Based on a one-second timer (323), the above command-response process (from sending a command 321 until screen update 322) is repeated at intervals of one second.

**[0104]** When an input from the user (clicking at a box) occurs, the box number that the user has chosen is assigned to variable D (324) and the terminal sends a command with parameter = D (321). The card receives the command (331) and compares the last generated random number R that has been stored with the received box number D (334). If a hit is found by this comparison, points are added and the result of the comparison is returned to the terminal (335). The terminal receives the result and shows the result on the screen (412). The above process is repeated by the number of chances to play (five times in the example shown in Fig. 17).

**[0105]** Fig. 19 shows the scripts description example for the shooting game illustrated in Figs. 17 and 18, described by using the scripts in Fig. 12. If the command parameter is "0." a random number is generated. Unless that, the user input is compared with the value of the last generated random number. Unlike the preceding examples of game scripts, this example of scripts includes a loop.

Lines 00 to 02: Work arrays are initialized.

Line 03: The beginning of loop

Line 04: Branching occurs, according to the value of the command parameter.

Lines 05 to 0d: When a value other than "0" is given as the command parameter, the user-input value is compared with the value of the random number and the result of comparison is stored. The loop counter increments by one. Unless the loop counter indicates the last time, the result of the comparison is returned to the terminal. After passing the processing control to the terminal, next time it returns, the processing returns to the beginning of loop (line 03). Lines 0e to 11: Following the last loop execution, points are added to the point count, based on the stored results of comparison.

Lines 12 to 16: If the command is a dummy, a random number is generated and returned to the terminal. The processing returns to the beginning of loop.

By repeating the above, the processing illustrated in Fig. 18 can be implemented.

**[0106]** As described above, by using the scripts given in Fig. 12, a plurality of types of games, such as three ones given as examples, can be defined. The foregoing three game scripts are examples and many types of games can be defined by describing the processes of generating a random number and comparing the random number with the input from the user in combination. The script interpreter executes these games defined in script descriptions as illustrated in Fig. 11, so that one application program can offer a plurality of types of games that can be selectively executed.

**[0107]** Although the above explanation discussed a game application program and its operation enabling a plurality of types of games to run on the smart card, this method can be applied to applications other than games. The present invention can be applied to application software for: e.g., a questionnaire system that the user can get points by answering questionnaires, which are updated after answered, shown on the screen; and an advertisement system that the user can get points by reading a specific advertisement. Such application generates value such as points in response to user operation and controls the contents so that the same suit of scripts will never be executed again. For this purpose, questionnaire answer logs and advertisement reading logs for each user must be stored in the card and control is required to prevent same scripts of questionnaire or

advertisement from being loaded into the card a plurality of times. For game scripts, once a suit of scripts has been executed, it is deleted or the rights to play game decrement by one, which is only required for script management. For questionnaire scripts, additional requirement is that the shop side collects the answer data.

**[0108]** Fig. 20 shows a questionnaire system operation example to which the present invention is applied. This example is essentially similar to the game system operation example illustrated in Fig. 10, except that the questionnaire answer data is collected.

**[0109]** A questionnaire application (500) is assumed to have been loaded into the smart card (110). This questionnaire application (500) comprises a command analyzer (211), a script interpreter (215), a processor for storing scripts of questionnaire (503), a processor for point data (214), and other components. Questionnaire data (501) is stored, each data thereof comprising a triple of entries: questionnaire scripts, issuer, and answer data. The answer data entry is blank before the user answers the questionnaire that is currently presented. When the user answers the questionnaire, the answer data is stored into place. This answer data entry also fills the role of a flag to indicate whether the user has answered the questionnaire. When this entry contains data, the same suit of questionnaire scripts cannot be executed again. Logs of answers to questionnaires are stored as the answer log (502) data to control script loading so that any questionnaire once answered will never be loaded again. As is the case in Fig. 9 and Fig. 10, point data (223) is stored as points managed per issuer Company.

**[0110]** At a shop (104) of Company (X), a terminal at shop (108) sends the smart card (110) a command to store questionnaire issued (506) and questionnaire data (507) containing questionnaire definition scripts and issuer. As is the case with game scripts, this data must be assured that it has been issued from the authorized issuer. For this purpose, a cipher key that authenticates the authorized issuer of the scripts, the key encrypted in accordance with the predefined scheme, must be attached to the questionnaire data. According to the cipher key, the processor for storing scripts of questionnaire (503) must perform authentication of the scripts it received. The processor for storing scripts of questionnaire (503) also checks to see whether the questionnaire data (507) it received has been answered by referring to the answer log (502) database, thereby performing the function of control over scripts to prevent answered questionnaire scripts from being stored into the questionnaire data (501) database.

**[0111]** When the user (113) answers a questionnaire at the user terminal (509), commands for answering (505) are sent to the smart card (110). If the questionnaire data (501) exists, the user can answer the questionnaire described in the questionnaire definition scripts. The questionnaire contents, not only requesting the user to answer straightforwardly, are elaborated so that the user can select the detail level of answer and the next questionnaire contents will change, according to the answer to the preceding one. Once the user has answered the questionnaire, points weighted in accordance with the answer are added to the point data (223) for the issuer of the questionnaire and the answer data is added to the questionnaire data (501) database in place coupled with the scripts. The questionnaire data (501) complete with the scripts and the answer data cannot be answered again. Adding answer data to the answered questionnaire data (501) is, in other words invalidating the questionnaire data.

**[0112]** When points are exchanged for a service at a terminal at shop (108) or the next questionnaire of the same issuer is issued, the answer data to the previously answered questionnaires must be collected. Because the answered questionnaire scripts coupled with answer data are retained in the invalid state, the questionnaire data (501) overflow is likely to occur if answer data is accumulated without being collected. Thus, collecting answer data at suitable timing is required. When a request for collecting answer data is issued to the card, the processor for collecting answer (504) encrypts answer data and returns it to the terminal (108). At the same time, the above processor erases the questionnaire data (501) for the scripts for which the answer data is lost and adds log information for the lost answer data to the answer log (502) database.

**[0113]** In the way described above, the application of the present invention can be expanded to cover the questionnaire system, not only applicable to games. Furthermore, the invention can be applied to the advertisement reading system in a similar way.

**[0114]** How the present invention which can be preferably embodied as explained above produces effects will be described in detail for each of the six major entities of the invention, specified in the section of "Means for Solving the Problems."

**[0115]** On the smart card that can be embodied as a preferred embodiment of the present invention with

(1) An application program comprising the following elements:

a) Game executing components
b) Storage for game definition scripts
c) Script interpreter
d) Storage and processor for point data
e) Storage and processor for rights to play game
f) Command analyzer

One application program installed on the smart card can offer a plurality of types of games that can be selectively executed.

In addition,

(2) Processor for storing game definition scripts

The processor enables the introduction of a new

type of game in any time even after the game application program is loaded into the smart card without being accompanied by application program exchange. Consequently, a more flexible game system can be provided.

The above processor controls storing game scripts, based on the game issue scheme prescribed in the application program, so that the processing parts of the application program can be changed in safety. This means the release from troublesome and time-consuming tasks accompanied by meeting the application issue scheme prescribed in the card OS.

Once the game application program for executing common processes has been loaded into smart cards on which different OSs are installed, such as "MULTOS" and "Java Card," programmers can program new games that are compatible on these cards without caring about the difference between the OSs.

(3) Function of point management per issuer enables the management of a plurality of types of games that are issued from a plurality of different issuers within one application. Consequently, a game system that is used more widely can be provided.

On the terminal device for operating with the smart card in question

(4) Function of issuing a game

(6) Function of calculating points

These functions make it possible that a plurality of types of games are issued to the smart card without being accompanied by game application program exchange and a flexible game system including point management is operated in safety.

Equipped with the user interface for playing games

(5) Function of executing a game

The function allows the user to enjoy playing with a game while the user can get points.

**[0116]** According to the present invention that can be preferably embodied as described herein, therefore, game applications for more flexible use can be introduced into a smart card system in safety. Increase of points linked with game results motivates the user to use the card more often and is effective for getting more customers from the viewpoint of the application provider. Furthermore, the present invention is significantly useful for making smart card systems popular.

**[0117]** Noticeable features of the present invention are the introduction of game definition scripts and a script interpreter and the method of control over storing scripts, based on the game issue scheme prescribed in the application program installed on the card. The invention provides a script loading/unloading method in which the contents of only the part for processing that is dependent on user input in a program, not limited to game application, is replaced by new contents without being accompanied by program exchange. This script loading/unloading method enables the development of more flexible and convenient smart card applications, which is released from platform restrictions.

**[0118]** The present invent makes it possible to replace game scripts in a program installed on a smart card by new game scripts without being accompanied by program exchange.

**[0119]** Technical items in connection with the present invention are listed below.

1. An application program that can run under an operating system (OS) installed on a smart card furnished with a storage means and an input/output interface and includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program.
2. The application program described in item 1, wherein the smart card includes point data storage for storing points the card user gained, the count of which may be updated by the result of program execution in accordance with the run procedure defined in scripts.
3. The application program described in item 1, wherein processing defined in scripts can generate different results, according to the input by the user of the smart card and the timing of execution thereof, and the user cannot predict the result of processing in advance.
4. The application program described in item 1, including a function that, following the execution of processing defined in scripts, invalidates those scripts and sets the processing impossible to do.
5. The application program described in item 1, including storage for rights to execute that stores the rights to execute scripts, defining the maximum number of times the processing defined in scripts can be executed, and a function that, immediately following the execution of processing defined in scripts, decrements the count of the rights to execute those scripts by one.
6. The application program described in item 1, configured such that scripts can be stored into the storage means through the input/output interface after the application program is loaded.
7. The application program described in item 1, including an authentication handler that performs a predetermined authentication procedure to assure that valid scripts, free of falsity, are stored when scripts are stored into the storage means through the input/output interface after the application program is loaded.
8. The application program described in item 2, including a function that adds up points per script issuer, attaches the identifier of issuer to the scripts or the rights to execute scripts, and just after processing defined in scripts is executed, updates

only the points associated with the issuer of the scripts, according to the result of the processing.

9. A storage medium holding an application program that can run under an operating system (OS) installed on a smart card furnished with a storage means and an input/output interface and includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program.

10. The storage medium described in item 9, wherein the smart card includes point data storage for storing points the card user gained, the count of which may be updated by the result of program execution in accordance with the run procedure defined in scripts.

11. The storage medium described in item 9, wherein processing defined in scripts can generate different results, according to the input by the user of the smart card and the timing of execution thereof, and the user cannot predict the result of processing in advance.

12. The storage medium described in item 9, including a function that, following the execution of processing defined in scripts, invalidates those scripts and sets the processing impossible to do.

13. The storage medium described in item 9, including storage for rights to execute that stores the rights to execute scripts, defining the maximum number of times the processing defined in scripts can be executed, and a function that, immediately following the execution of processing defined in scripts, decrements the count of the rights to execute those scripts by one.

14. The storage medium described in item 9, configured such that scripts can be stored into the storage means through the input/output interface after the application program is loaded.

15. The storage medium described in item 9, including an authentication handler that performs a predetermined authentication procedure to assure that valid scripts, free of falsity, are stored when scripts are stored into the storage means through the input/output interface after the application program is loaded.

16. The storage medium described in item 10, including a function that adds up points per script issuer, attaches the identifier of issuer to the scripts or the rights to execute scripts, and just after processing defined in scripts is executed; updates only the points associated with the issuer of the scripts, according to the result of the processing.

17. A terminal device capable of operating with a smart card furnished with a storage means and an input/output interface, including a means to load an application program that can run under an operating system (OS) installed on the smart card and includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program into the smart card via the input/output interface.

18. A terminal device capable of operating with a smart card furnished with a storage means and an input/output interface, including a means to load scripts as part of an application program from outside via the input/output interface into the smart card wherein the application program that can run under an operating system (OS) installed on the smart card and includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program is stored, in the storage means.

**[0120]** As many apparently widely different embodiments of this invention may be made without departing from the spirit and scope thereof, it is to be understood that the invention is not limited to the specific embodiments thereof except as defined in the appended claims.

## Claims

1. A method of loading an application program into a smart card furnished with a storage means and an input/output interface, whereby the application program that can run under an operating system (OS) installed on the smart card and includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program is loaded from outside via the input/output interface into the smart card.

2. The method of loading an application program into a smart card according to claim 1, wherein the application program is configured such that common process components as a set of a plurality of processing modules each of which outputs a given result in response to a given command are packaged inside the program in order that the processing modules in the common process components are selectively called when the scripts are interpreted and executed.

3. The method of loading an application program into a smart card according to claim 1, wherein the scripts are installed into the smart card after the application program is installed into the smart card.

4. The method of loading an application program into a smart card according to claim 1, wherein the scripts are stored into the storage means through the input/output interface after the application program is loaded.

5. The method of loading an application program into a smart card according to claim 1, wherein a first

script is installed into the smart card after the application program is installed into the smart card and a second script that is different type from the first script is installed into the smart card.

6. The method of loading an application program into a smart card according to claim 1, wherein processing defined in the scripts can generate different results, according to the input by the user of the smart card and the timing of execution thereof, and the user cannot predict the result of processing in advance.

7. The method of loading an application program into a smart card according to claim 1, including a process that, following the execution of processing defined in the scripts, invalidates those scripts and makes the processing impossible to do.

8. The method of loading an application program into a smart card according to claim 1, including storage for rights to execute that stores the rights to execute scripts, defining the maximum number of times the processing defined in the scripts can be executed, and a function that, immediately following the execution of processing defined in the scripts, decrements the count of the rights to execute those scripts by one.

9. The method of loading an application program into a smart card according to claim 1, wherein a predetermined authentication procedure is carried out to assure that valid scripts, free of falsity, are stored when the scripts are stored into the storage means through the input/output interface after the application program is loaded.

10. The method of loading an application program into a smart card according to claim 1, wherein the scripts define a game and by the execution of a suit of scripts by using the smart card on which the script suit has been installed, the game defined in the script suit can be executed.

11. The method of loading an application program into a smart card according to claim 1, wherein the scripts define a game, points may be added, according to the result of executing the game, and the accumulated points are retained in the smart card.

12. The method of loading an application program into a smart card according to claim 1, wherein the scripts define a game, the game can be executed only if rights to play with the game are given to the card user, and the rights to play define what number of times a type of game can be executed and are stored into the smart card when the game is issued.

13. The method of loading an application program into a smart card according to claim 11, wherein data of the rights to play game may be stored separately from the scripts or the right(s) to play game may be included in the game scripts; in the latter case, the right(s) to play will be lost, following the execution of the game.

14. The method of loading an application program into a smart card according to claim 11, wherein the data of rights to play game includes the script issuer that offers the point service, the type of game playable, and the count of games that the user can play with that game.

15. The method of loading an application program into a smart card according to claim 11, wherein the point data is stored per script issuer that offers the point service.

16. A method of loading scripts into a smart card furnished with a storage means and an input/output interface, whereby the scripts are loaded as part of an application program from outside via the input/output interface into the smart card wherein the application program that can run under an operating system (OS) installed on the smart card and includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program is stored in the storage means.

17. A smart card furnished with a storage means and an input/output interface, holding an application program that can run under an operating system (OS) installed on the smart card, includes an interpreter for interpreting and executing scripts described to define the run procedure of the application program, and is loaded from outside via the input/output interface into the smart card.

FIG. 1

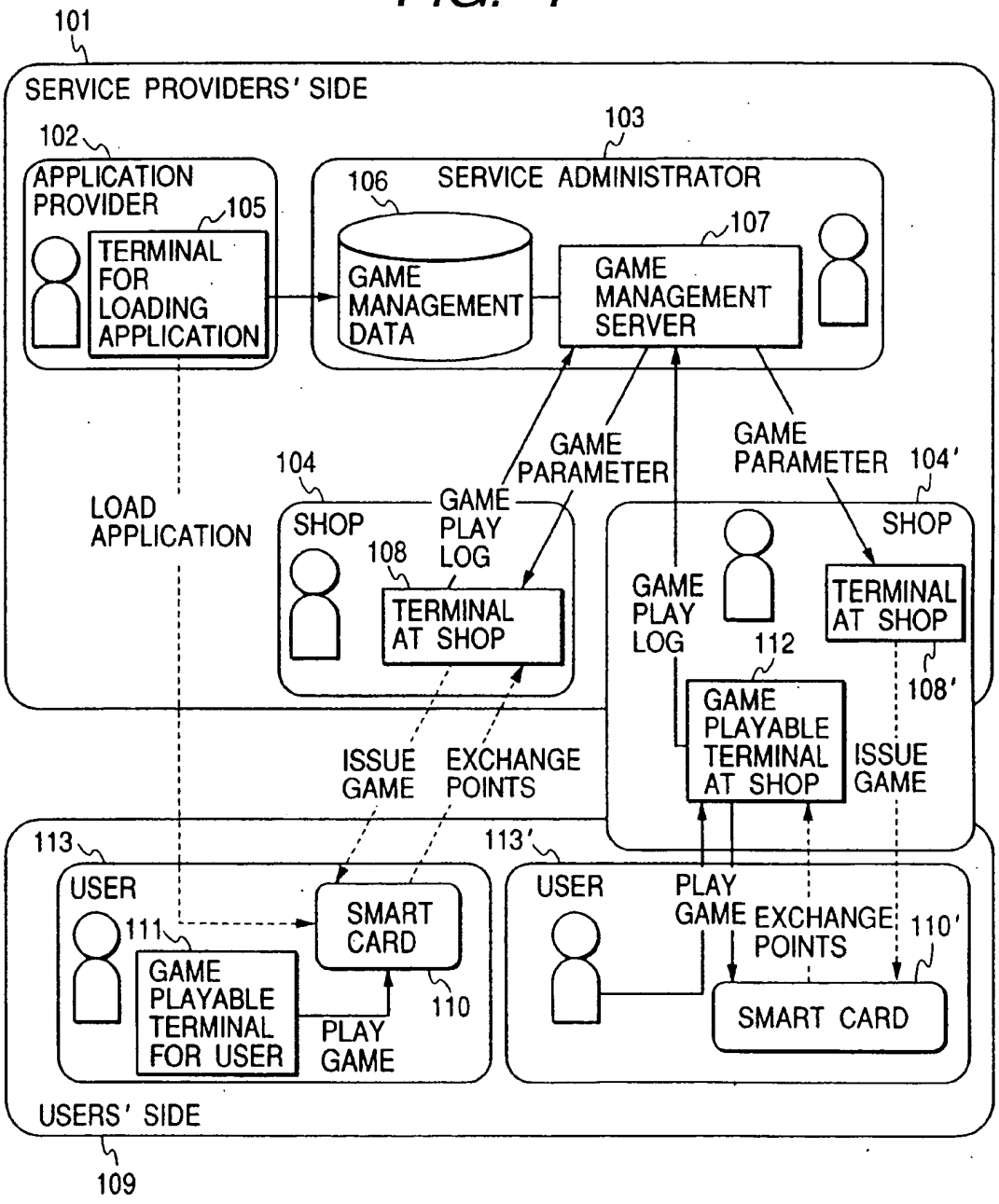

101
SERVICE PROVIDERS' SIDE
102
APPLICATION PROVIDER
105
TERMINAL FOR LOADING APPLICATION
103
106
SERVICE ADMINISTRATOR
GAME MANAGEMENT DATA
107
GAME MANAGEMENT SERVER
LOAD APPLICATION
104
SHOP
GAME PLAY LOG
108
TERMINAL AT SHOP
GAME PARAMETER
GAME PARAMETER
104'
SHOP
GAME PLAY LOG
112
GAME PLAYABLE TERMINAL AT SHOP
TERMINAL AT SHOP
108'
ISSUE GAME
EXCHANGE POINTS
ISSUE GAME
113
USER
111
GAME PLAYABLE TERMINAL FOR USER
SMART CARD
110
PLAY GAME
113'
USER
PLAY GAME
EXCHANGE POINTS
110'
SMART CARD
USERS' SIDE
109

FIG. 2

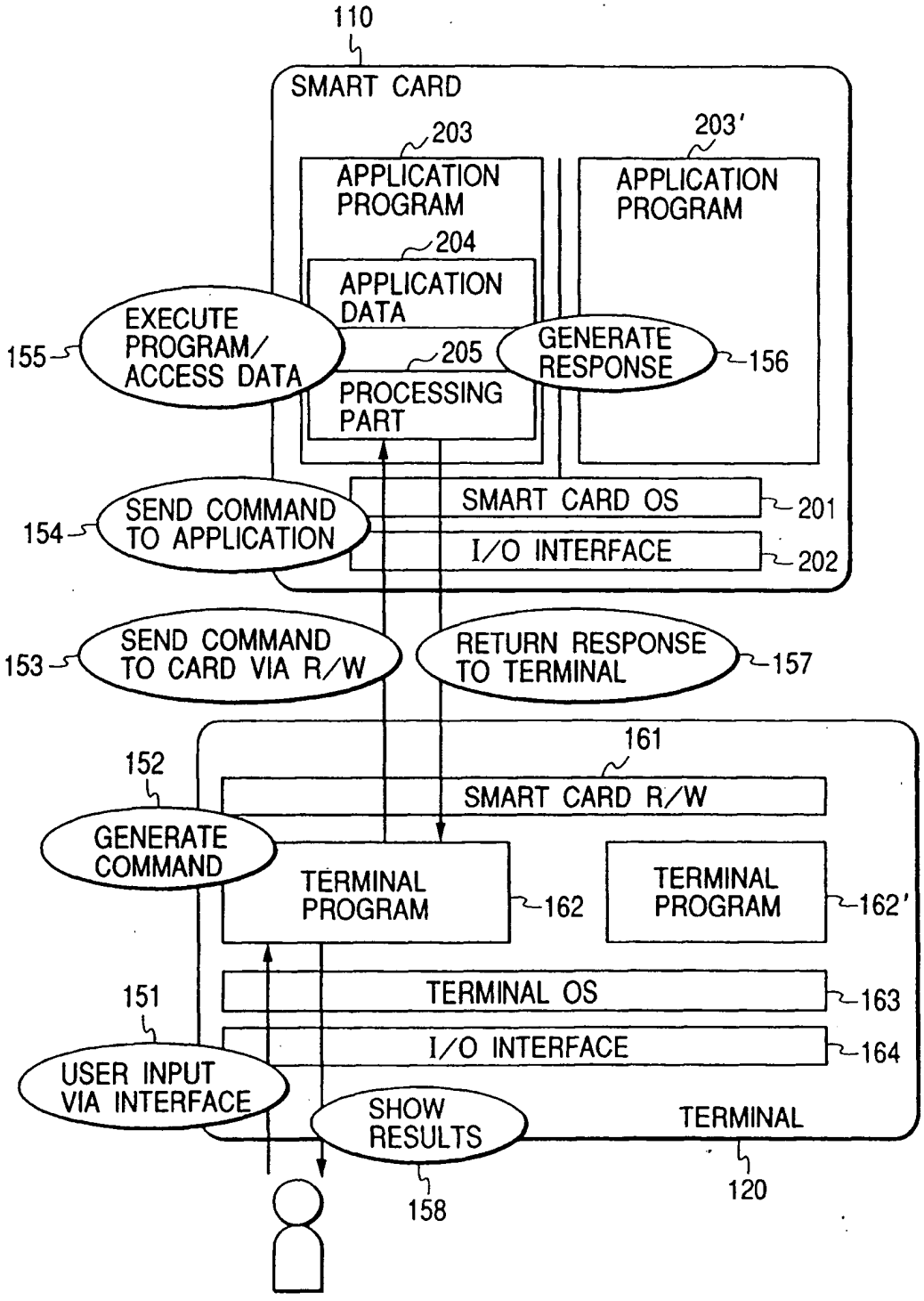

110
SMART CARD
203
APPLICATION PROGRAM
204
APPLICATION DATA
205
PROCESSING PART
203'
APPLICATION PROGRAM
155
EXECUTE PROGRAM/ ACCESS DATA
GENERATE RESPONSE
156
154
SEND COMMAND TO APPLICATION
SMART CARD OS
201
I/O INTERFACE
202
153
SEND COMMAND TO CARD VIA R/W
RETURN RESPONSE TO TERMINAL
157
161
152
SMART CARD R/W
GENERATE COMMAND
TERMINAL PROGRAM
162
TERMINAL PROGRAM
162'
151
TERMINAL OS
163
I/O INTERFACE
164
USER INPUT VIA INTERFACE
SHOW RESULTS
158
TERMINAL
120

FIG. 3

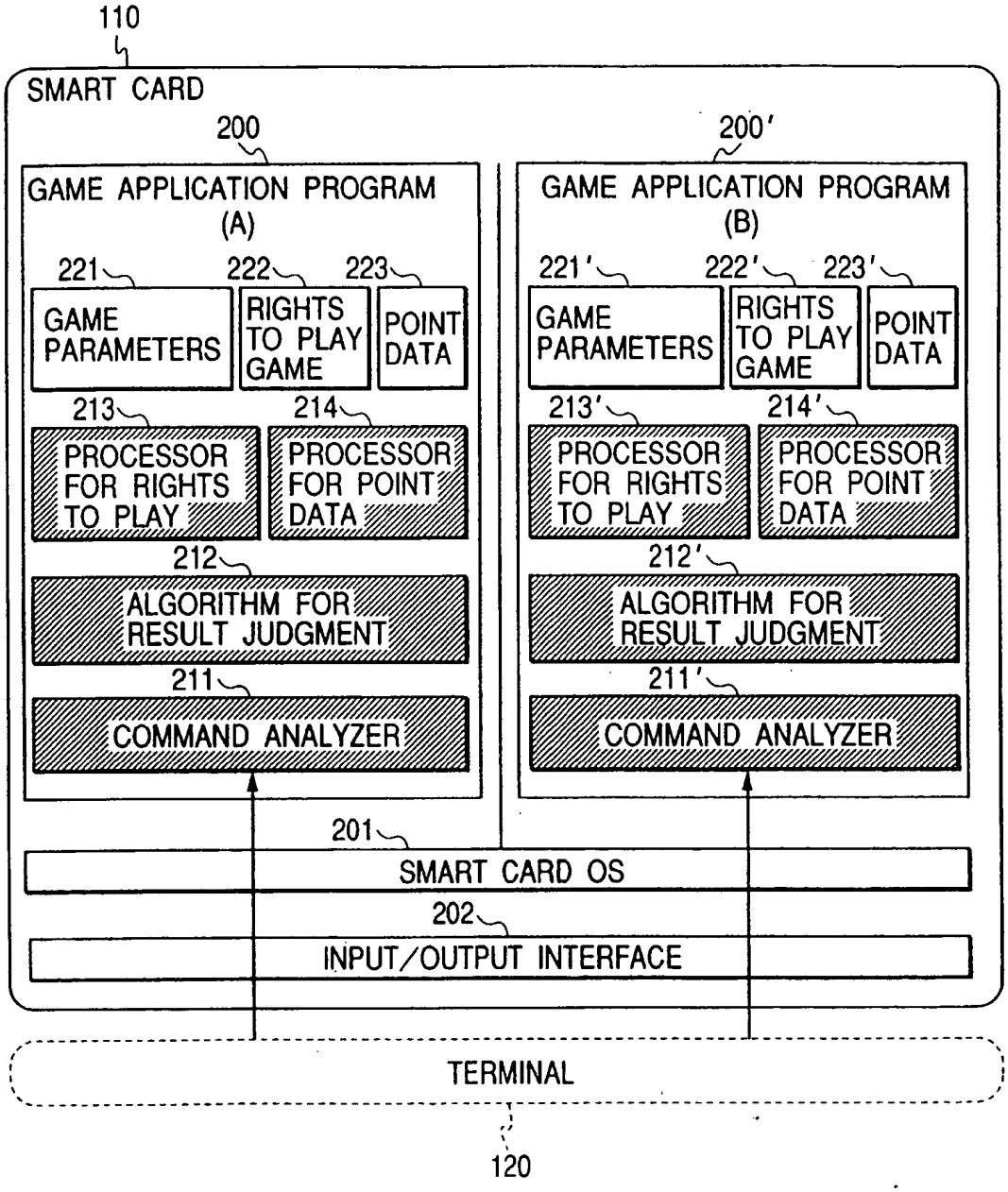

110
SMART CARD
200
GAME APPLICATION PROGRAM (A)
221
GAME PARAMETERS
222
RIGHTS TO PLAY GAME
223
POINT DATA
213
PROCESSOR FOR RIGHTS TO PLAY
214
PROCESSOR FOR POINT DATA
212
ALGORITHM FOR RESULT JUDGMENT
211
COMMAND ANALYZER
200'
GAME APPLICATION PROGRAM (B)
221'
GAME PARAMETERS
222'
RIGHTS TO PLAY GAME
223'
POINT DATA
213'
PROCESSOR FOR RIGHTS TO PLAY
214'
PROCESSOR FOR POINT DATA
212'
ALGORITHM FOR RESULT JUDGMENT
211'
COMMAND ANALYZER
201
SMART CARD OS
202
INPUT/OUTPUT INTERFACE
TERMINAL
120

FIG. 4

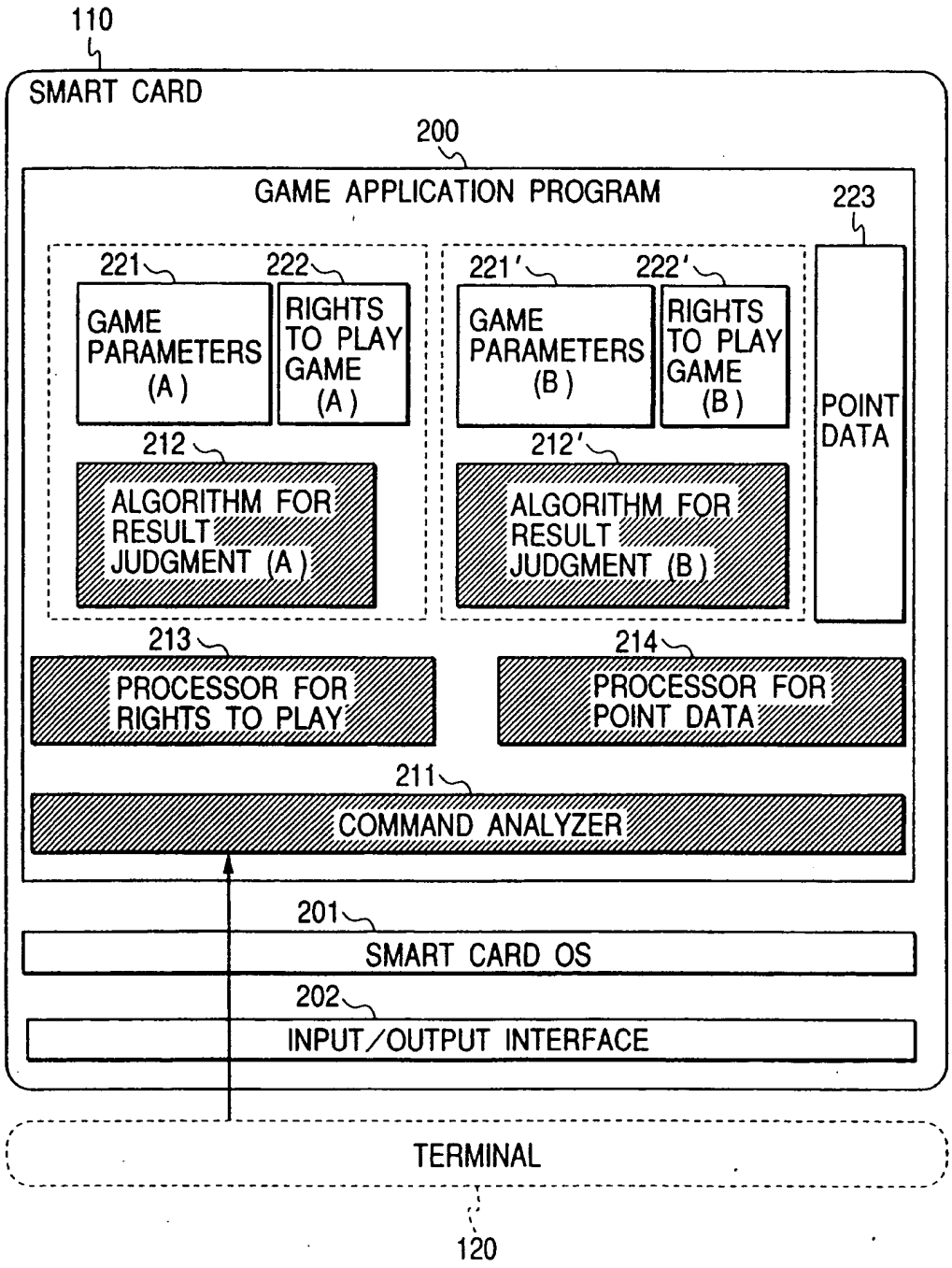

110
SMART CARD
200
GAME APPLICATION PROGRAM
223
221
222
GAME PARAMETERS (A)
RIGHTS TO PLAY GAME (A)
221'
222'
GAME PARAMETERS (B)
RIGHTS TO PLAY GAME (B)
POINT DATA
212
ALGORITHM FOR RESULT JUDGMENT (A)
212'
ALGORITHM FOR RESULT JUDGMENT (B)
213
PROCESSOR FOR RIGHTS TO PLAY
214
PROCESSOR FOR POINT DATA
211
COMMAND ANALYZER
201
SMART CARD OS
202
INPUT/OUTPUT INTERFACE
TERMINAL
120

FIG. 5

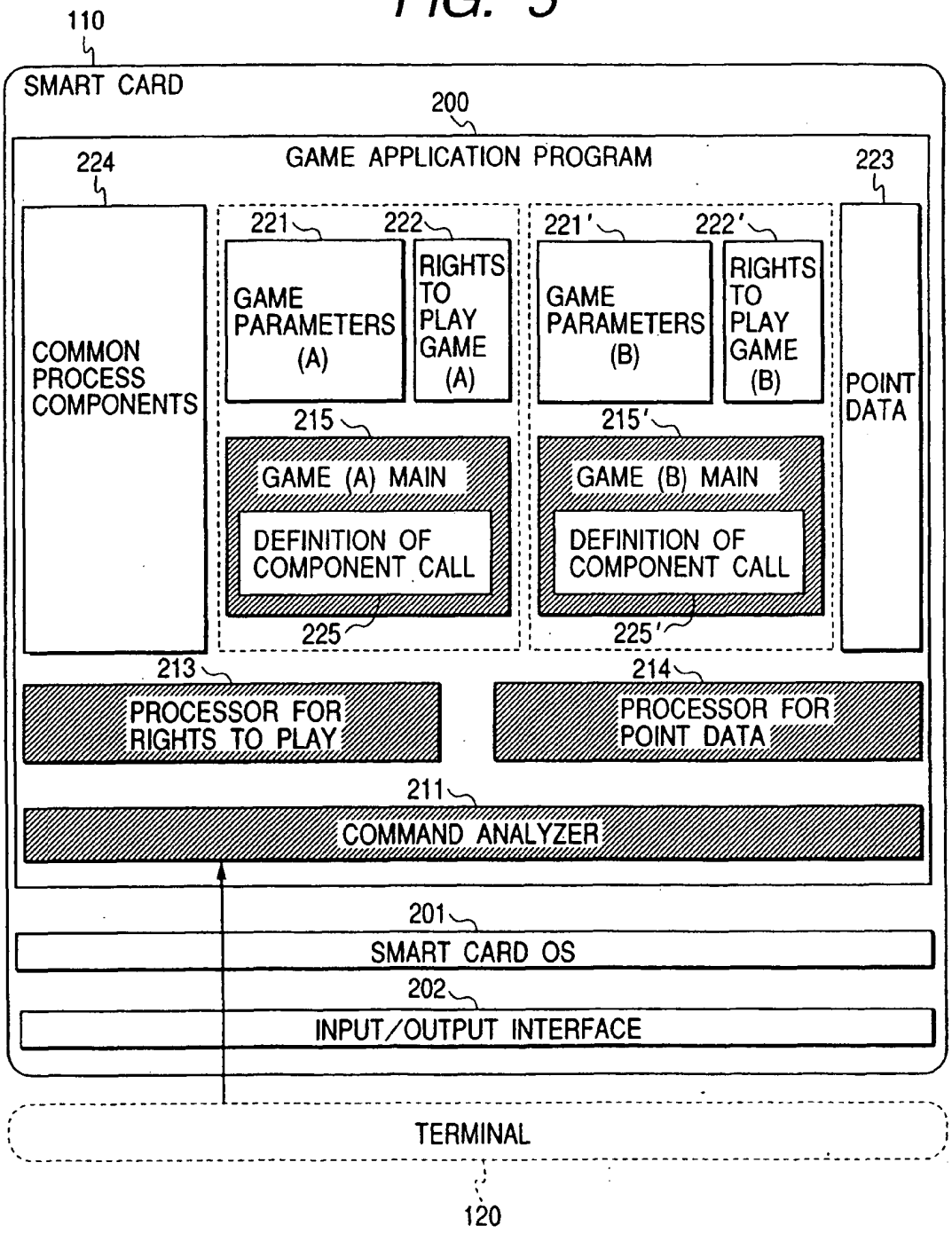

110
SMART CARD
200
GAME APPLICATION PROGRAM
224
223
COMMON PROCESS COMPONENTS
221
222
GAME PARAMETERS (A)
RIGHTS TO PLAY GAME (A)
221'
222'
GAME PARAMETERS (B)
RIGHTS TO PLAY GAME (B)
POINT DATA
215
GAME (A) MAIN
DEFINITION OF COMPONENT CALL
225
215'
GAME (B) MAIN
DEFINITION OF COMPONENT CALL
225'
213
PROCESSOR FOR RIGHTS TO PLAY
214
PROCESSOR FOR POINT DATA
211
COMMAND ANALYZER
201
SMART CARD OS
202
INPUT/OUTPUT INTERFACE
TERMINAL
120

FIG. 6

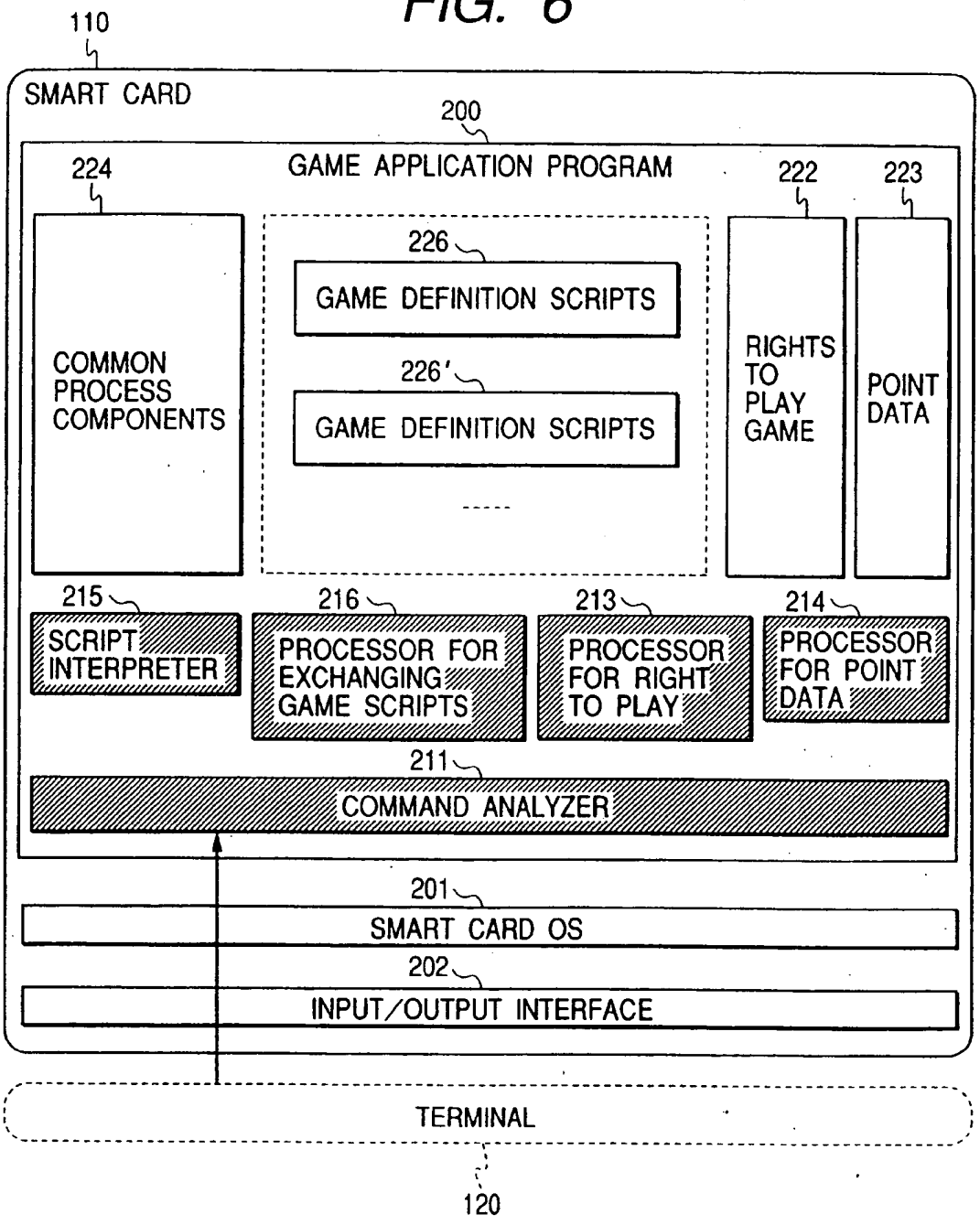

110
SMART CARD
200
GAME APPLICATION PROGRAM
224
222
223
226
GAME DEFINITION SCRIPTS
226'
GAME DEFINITION SCRIPTS
COMMON PROCESS COMPONENTS
RIGHTS TO PLAY GAME
POINT DATA
215
216
213
214
SCRIPT INTERPRETER
PROCESSOR FOR EXCHANGING GAME SCRIPTS
PROCESSOR FOR RIGHT TO PLAY
PROCESSOR FOR POINT DATA
211
COMMAND ANALYZER
201
SMART CARD OS
202
INPUT/OUTPUT INTERFACE
TERMINAL
120

FIG. 7

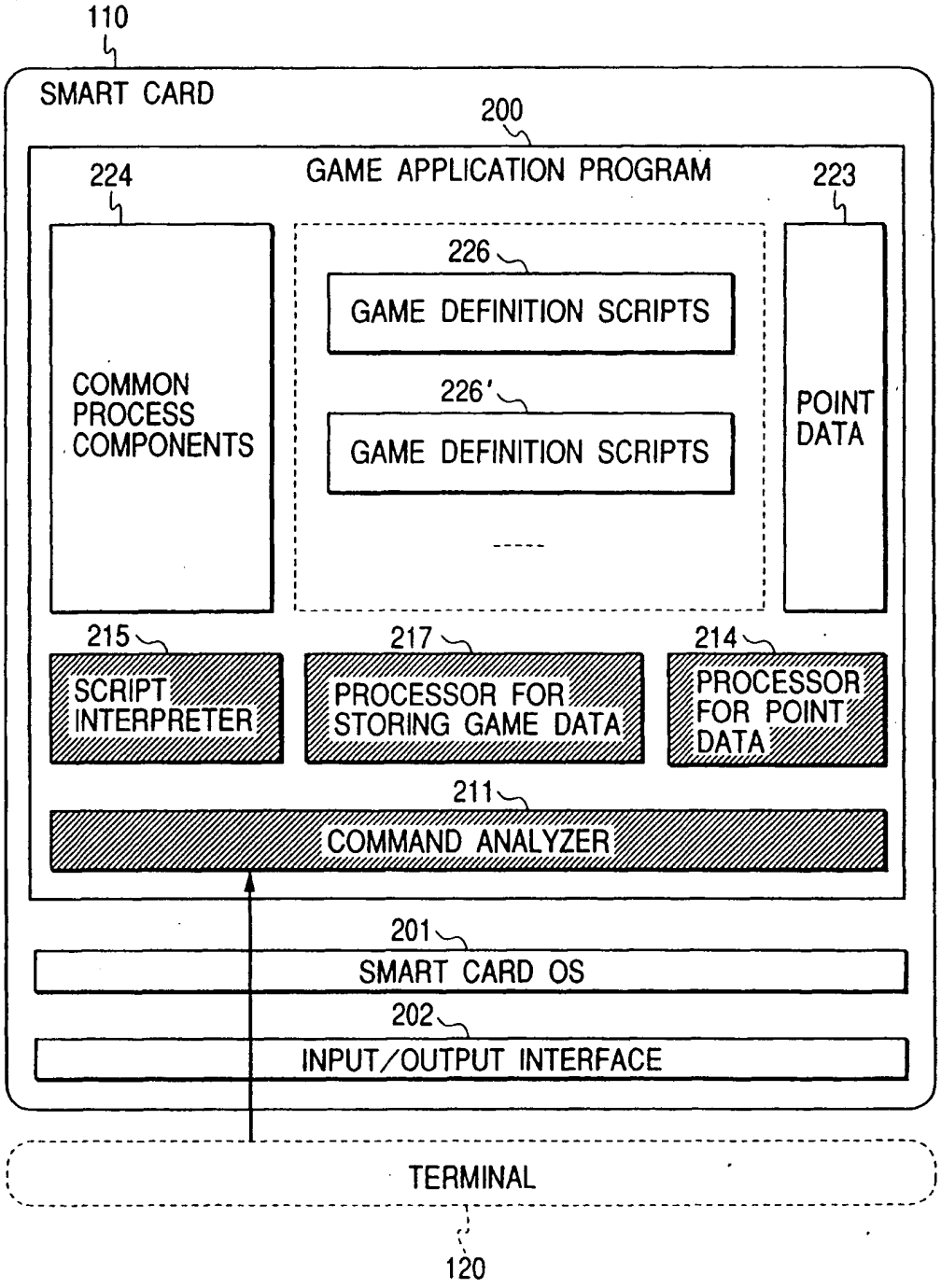

110
SMART CARD
200
GAME APPLICATION PROGRAM
224
223
226
GAME DEFINITION SCRIPTS
226'
GAME DEFINITION SCRIPTS
COMMON PROCESS COMPONENTS
POINT DATA
215
217
214
SCRIPT INTERPRETER
PROCESSOR FOR STORING GAME DATA
PROCESSOR FOR POINT DATA
211
COMMAND ANALYZER
201
SMART CARD OS
202
INPUT/OUTPUT INTERFACE
TERMINAL
120

FIG. 8

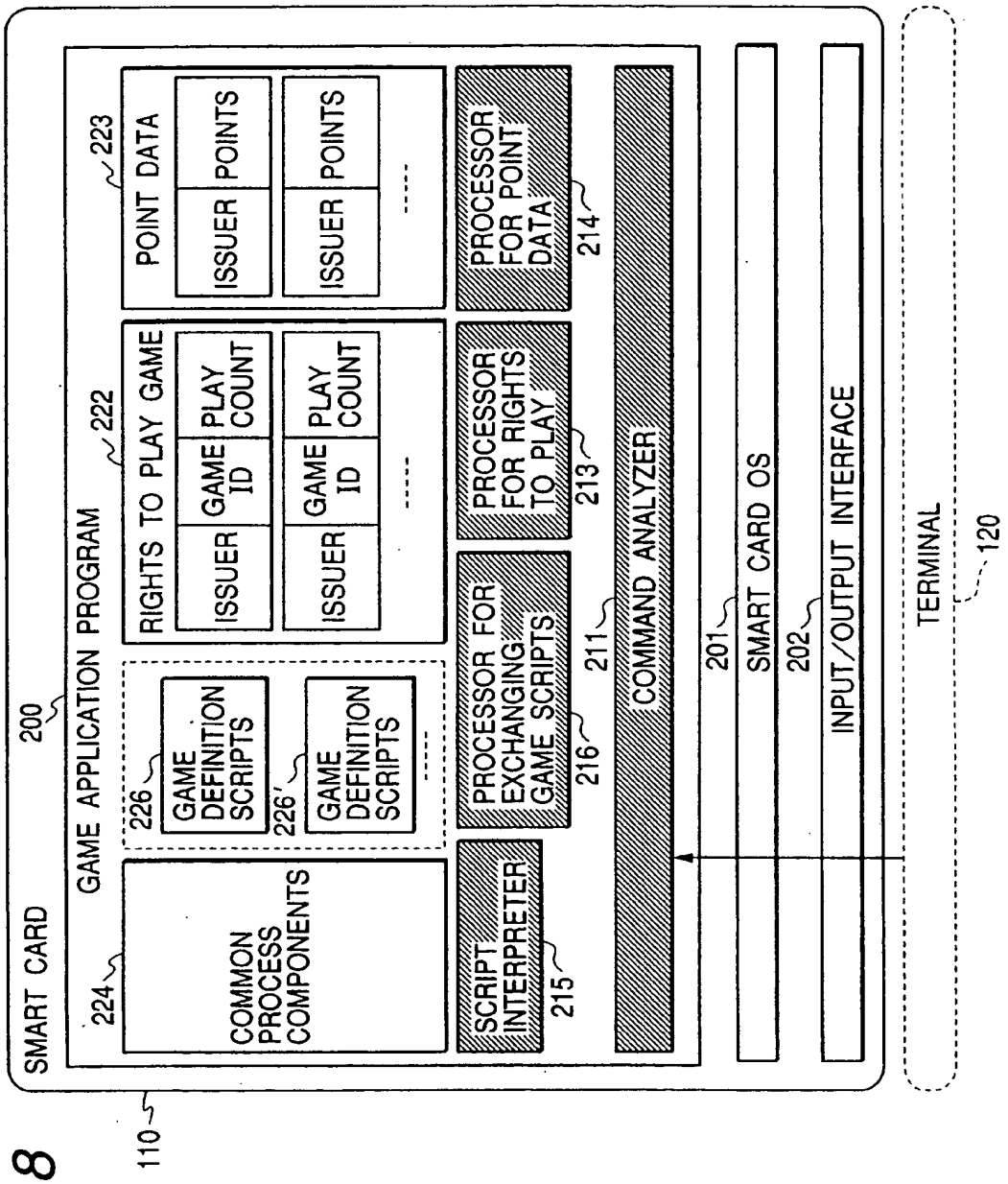

SMART CARD
110
GAME APPLICATION PROGRAM
200
COMMON PROCESS COMPONENTS
224
GAME DEFINITION SCRIPTS
226
GAME DEFINITION SCRIPTS
226'
RIGHTS TO PLAY GAME
222
ISSUER
GAME ID
PLAY COUNT
POINT DATA
223
ISSUER
POINTS
SCRIPT INTERPRETER
215
PROCESSOR FOR EXCHANGING GAME SCRIPTS
216
PROCESSOR FOR RIGHTS TO PLAY
213
PROCESSOR FOR POINT DATA
214
COMMAND ANALYZER
211
SMART CARD OS
201
INPUT/OUTPUT INTERFACE
202
TERMINAL
120

FIG. 9

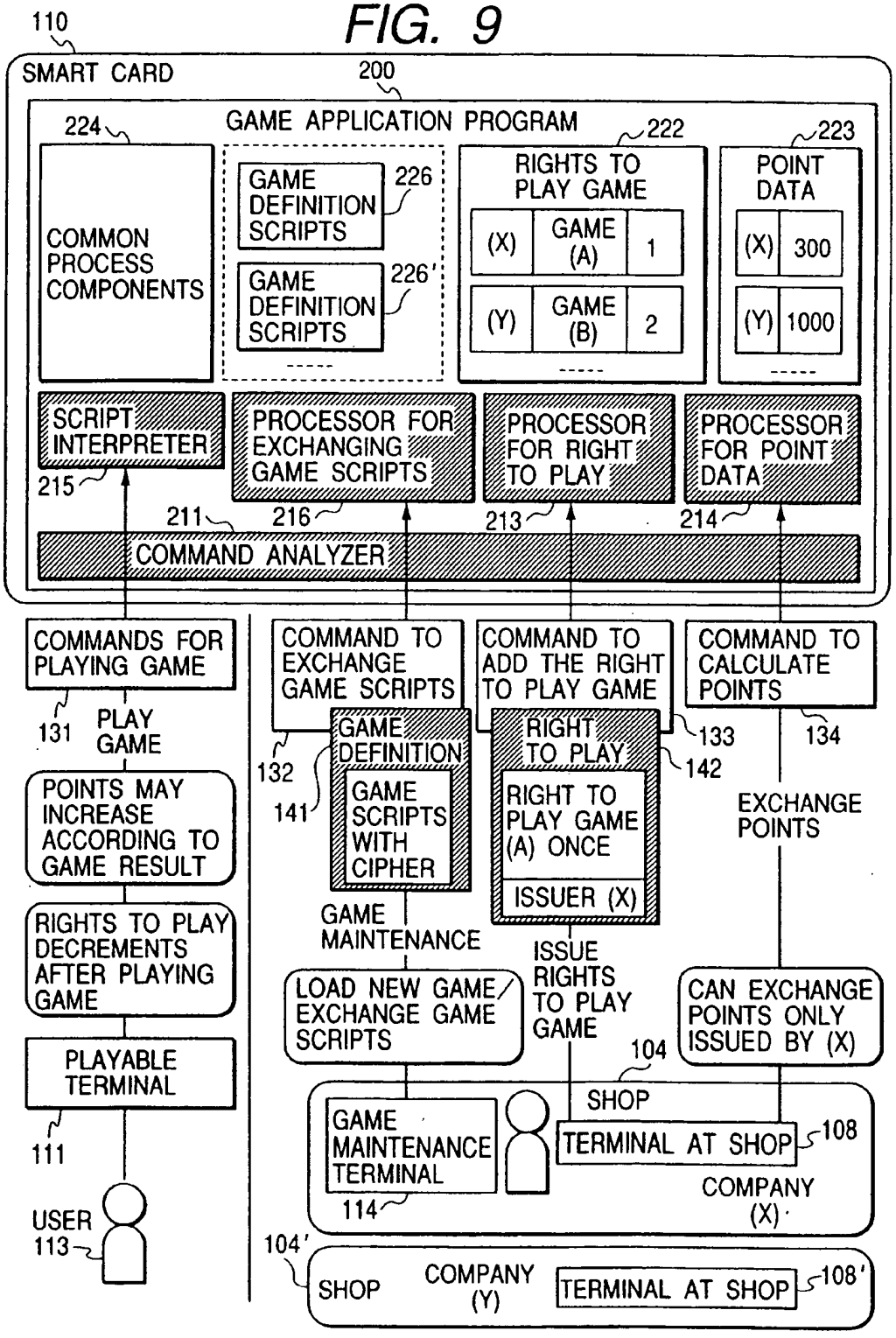

110
SMART CARD
200
GAME APPLICATION PROGRAM
224
COMMON PROCESS COMPONENTS
GAME DEFINITION SCRIPTS
226
GAME DEFINITION SCRIPTS
226'
222
RIGHTS TO PLAY GAME
(X) GAME (A) 1
(Y) GAME (B) 2
223
POINT DATA
(X) 300
(Y) 1000
SCRIPT INTERPRETER
215
PROCESSOR FOR EXCHANGING GAME SCRIPTS
PROCESSOR FOR RIGHT TO PLAY
PROCESSOR FOR POINT DATA
211
216
213
214
COMMAND ANALYZER
COMMANDS FOR PLAYING GAME
131
PLAY GAME
POINTS MAY INCREASE ACCORDING TO GAME RESULT
RIGHTS TO PLAY DECREMENTS AFTER PLAYING GAME
PLAYABLE TERMINAL
111
USER
113
COMMAND TO EXCHANGE GAME SCRIPTS
132
GAME DEFINITION
141
GAME SCRIPTS WITH CIPHER
GAME MAINTENANCE
LOAD NEW GAME/ EXCHANGE GAME SCRIPTS
COMMAND TO ADD THE RIGHT TO PLAY GAME
133
RIGHT TO PLAY
142
RIGHT TO PLAY GAME (A) ONCE
ISSUER (X)
ISSUE RIGHTS TO PLAY GAME
COMMAND TO CALCULATE POINTS
134
EXCHANGE POINTS
CAN EXCHANGE POINTS ONLY ISSUED BY (X)
104
SHOP
GAME MAINTENANCE TERMINAL
114
TERMINAL AT SHOP
108
COMPANY (X)
104'
SHOP
COMPANY (Y)
TERMINAL AT SHOP
108'

FIG. 10

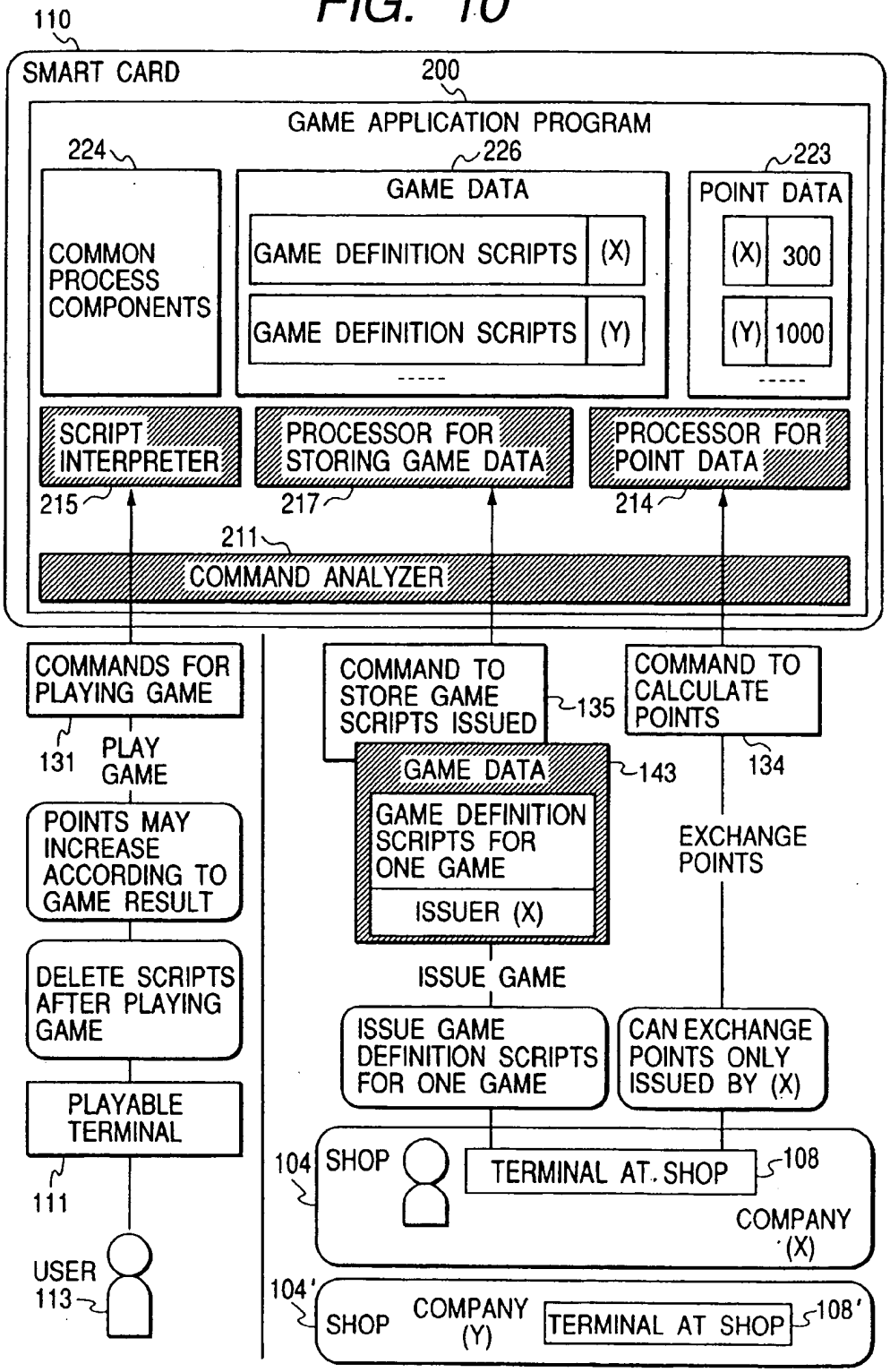
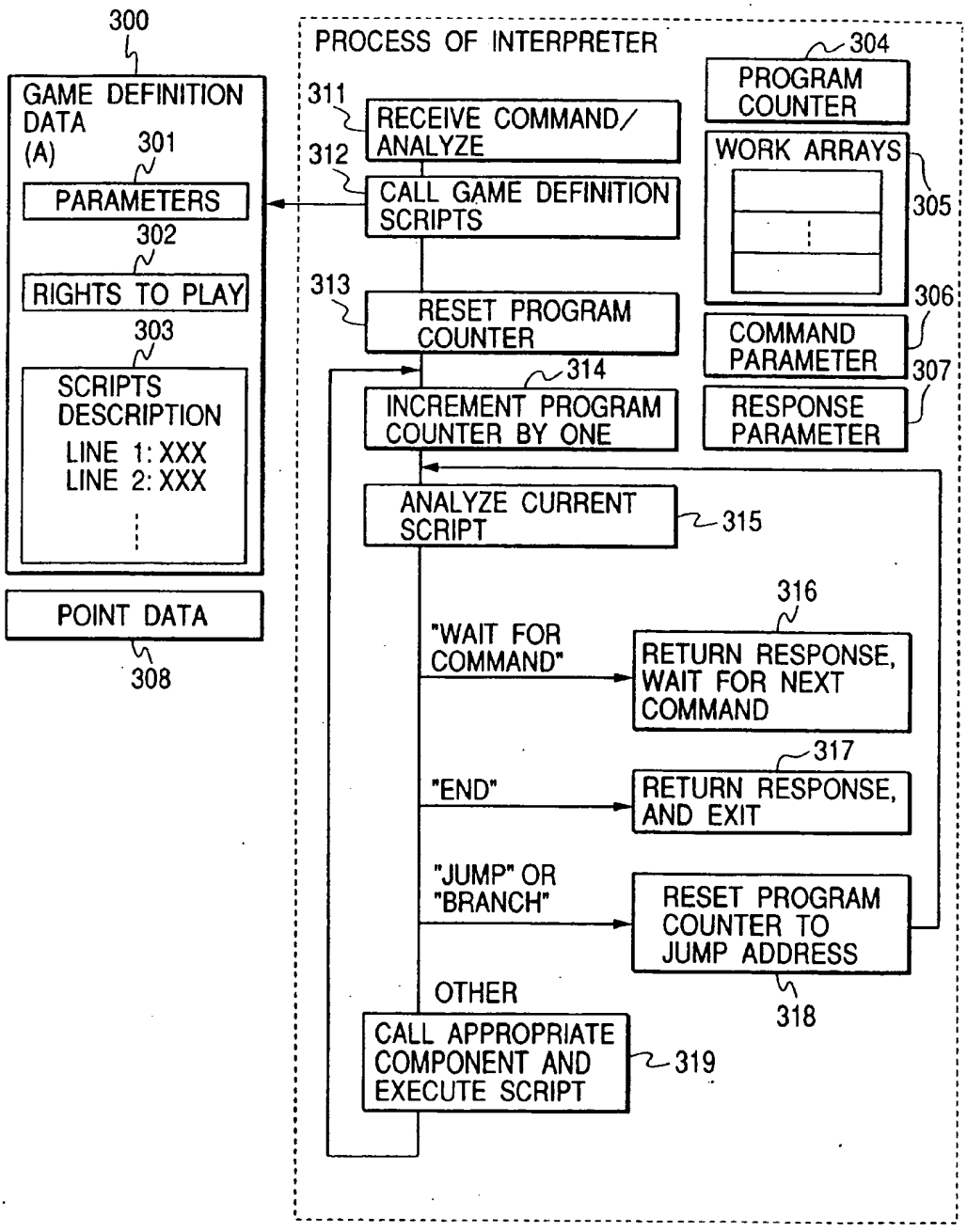

110
SMART CARD
200
GAME APPLICATION PROGRAM
224
COMMON PROCESS COMPONENTS
226
GAME DATA
GAME DEFINITION SCRIPTS (X)
GAME DEFINITION SCRIPTS (Y)
223
POINT DATA
(X) 300
(Y) 1000
SCRIPT INTERPRETER
PROCESSOR FOR STORING GAME DATA
PROCESSOR FOR POINT DATA
215
217
214
211
COMMAND ANALYZER
COMMANDS FOR PLAYING GAME
131
PLAY GAME
POINTS MAY INCREASE ACCORDING TO GAME RESULT
DELETE SCRIPTS AFTER PLAYING GAME
PLAYABLE TERMINAL
111
USER
113
COMMAND TO STORE GAME SCRIPTS ISSUED
135
GAME DATA
143
GAME DEFINITION SCRIPTS FOR ONE GAME
ISSUER (X)
ISSUE GAME
ISSUE GAME DEFINITION SCRIPTS FOR ONE GAME
COMMAND TO CALCULATE POINTS
134
EXCHANGE POINTS
CAN EXCHANGE POINTS ONLY ISSUED BY (X)
104
SHOP
TERMINAL AT SHOP
108
COMPANY (X)
104'
SHOP
COMPANY (Y)
TERMINAL AT SHOP
108'

FIG. 11


300
GAME DEFINITION DATA (A)
301
PARAMETERS
302
RIGHTS TO PLAY
303
SCRIPTS DESCRIPTION
LINE 1: XXX
LINE 2: XXX
POINT DATA
308
PROCESS OF INTERPRETER
304
PROGRAM COUNTER
WORK ARRAYS
305
306
COMMAND PARAMETER
307
RESPONSE PARAMETER
311
RECEIVE COMMAND/ ANALYZE
312
CALL GAME DEFINITION SCRIPTS
313
RESET PROGRAM COUNTER
314
INCREMENT PROGRAM COUNTER BY ONE
ANALYZE CURRENT SCRIPT
315
316
"WAIT FOR COMMAND"
RETURN RESPONSE, WAIT FOR NEXT COMMAND
317
"END"
RETURN RESPONSE, AND EXIT
"JUMP" OR "BRANCH"
RESET PROGRAM COUNTER TO JUMP ADDRESS
318
OTHER
CALL APPROPRIATE COMPONENT AND EXECUTE SCRIPT
319

# FIG. 12

※ array[]: work array (305)
cmd[] : command parameter (306)
rsp[] : response parameter (307)

[store data]

| | |
|---|---|
| set (a, b) | Store value b into array[a] |
| random (a, b) | Generate a random number (0 to b) and store it into array[a] |
| getcmd (a, b) | Store value of cmd[b] into array[a] |
| setrspa (a, b) | Store the contents of array[a] into rsp[b] |
| setrspv (a, b) | Store value a into rsp[b] |
| getcmda (a, b) | Store the contents of array[(cmd[b])] into array[a] |

[addition/subtraction and compare]

| | |
|---|---|
| adda (a, b, c) | Add the contents of array[b] and the contents of array[c] and store the result into array[a] |
| suba (a, b, c) | Subtract the contents of array[c] from the contents of array[b] and store the result into array[a] |
| cmpa (a, b, c) | Compare array[b] and array[c] contents and store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy<zz) |
| addv (a, b, c) | Add the contents of array[b] and the value of c and store the result into array[a] |
| subv (a, b, c) | Subtract the value of c from the contents of array[b] and store the result into array[a] |
| cmpv (a, b, c) | Compare the contents of array[b] and the value of c and store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy<zz) |

[jump, branch]

| | |
|---|---|
| jmp (a) | Jump to line a |
| eq (a, b, c) | Jump to line c if the contents of array[a]= the value of b |
| ne (a, b, c) | Jump to line c if the contents of array[a]≠ the value of b |

[point calculation]

| | |
|---|---|
| pointa (a) | Add the contents of array[a] to point data |
| pointv (a) | Add value a to point data |

[return response to terminal]

| | |
|---|---|
| return | Return rsp[] as response data, and wait for next command |
| end | Return rsp[] as response data, and exit from game script execution |

FIG. 13

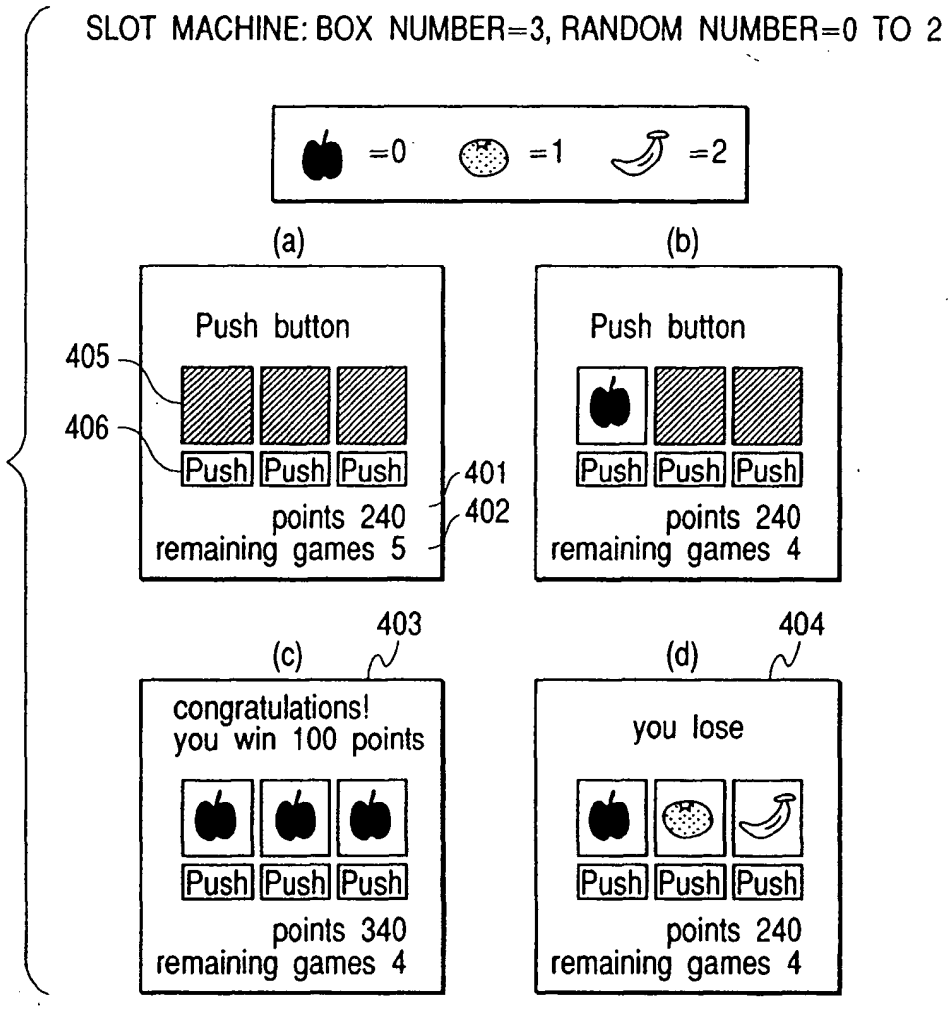

SLOT MACHINE: BOX NUMBER=3, RANDOM NUMBER=0 TO 2
=0
=1
=2
(a)
(b)
Push button
Push button
405
406
Push Push Push
Push Push Push
401
402
points 240
remaining games 5
points 240
remaining games 4
403
404
(c)
(d)
congratulations!
you win 100 points
you lose
Push Push Push
Push Push Push
points 340
remaining games 4
points 240
remaining games 4

FIG. 14

| LINE | SCRIPTS | COMMENT |
|---|---|---|
| 00 | random (0, 2) | Store a random number (0 to 2) into array[0] |
| 01 | setrspa (0, 0) | Store the contents of array[0] into rsp[0] |
| 02 | return | Return response, and wait for command |
| 03 | random (1, 2) | Store a random number (0 to 2) into array[1] |
| 04 | setrspa (1, 0) | Store the contents of array[1] to rsp[0] |
| 05 | return | Return response, and wait for command |
| 06 | random (2, 2) | Store a random number (0 to 2) into array[2] |
| 07 | setrspa (2, 0) | Store the contents of array[2] into rsp[0] |
| 08 | setrspv (1, 1) | Store a value of 1 (to indicate the lost) into rsp[1] |
| 09 | cmpa (3, 0, 1) | Compare array[0] and array[1] contents and store the result into array[3] |
| 0a | cmpa (4, 0, 2) | Compare array[0] and array[2] contents and store the result into array[4] |
| 0b | eq (3, 0, 0d) | Jump to line 0×0d if array[3] contains 0 |
| 0c | end | Return response, and exit |
| 0d | eq (4, 0, 0f) | Jump to line 0×0f if array[4] contains 0 |
| 0e | end | Return response, and exit |
| 0f | pointa (50) | Add 50 to point data |
| 10 | setrspv (2, 50) | Store 50 (points won at this game) into rsp[2] |
| 11 | end | Return response, and exit |

FIG. 15

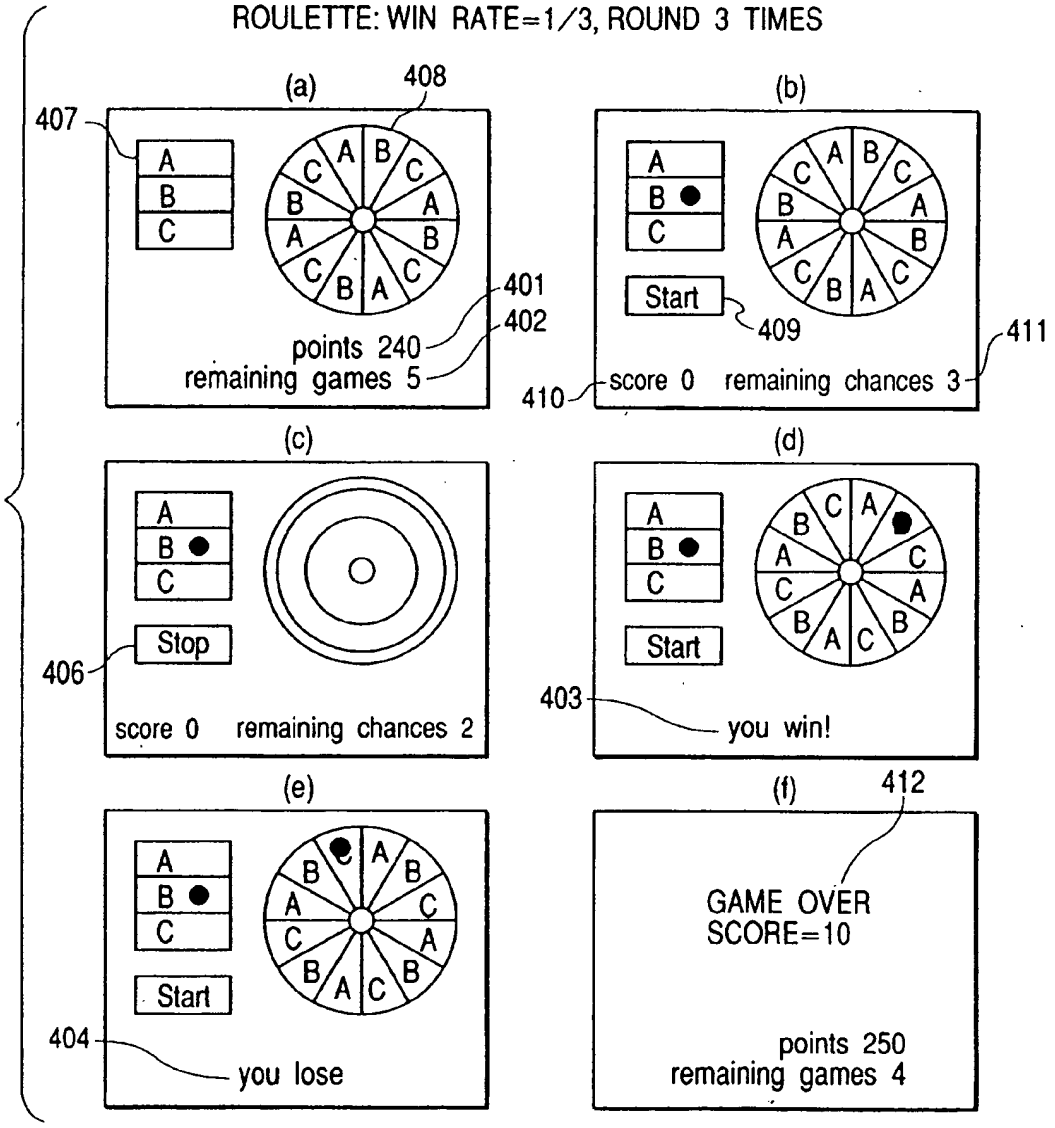

ROULETTE: WIN RATE=1/3, ROUND 3 TIMES
(a)
408
407
A
B
C
401
402
points 240
remaining games 5
(b)
Start
409
411
410
score 0 remaining chances 3
(c)
Stop
406
score 0 remaining chances 2
(d)
Start
403
you win!
(e)
Start
404
you lose
(f)
412
GAME OVER
SCORE=10
points 250
remaining games 4

FIG. 16

| LINE | SCRIPT | COMMENT |
|---|---|---|
| 00 | getcmd (0, 0) | Store the value of cmd[0] into array[0] |
| 01 | random (1, 2) | Store a random number (0 to 2) into array[1] |
| 02 | cmpa (2, 0, 1) | Compare array[0] and array[1] contents and store the result into array[2] |
| 03 | setrspa (1, 0) | Store the contents of array[1] into rsp[0] |
| 04 | setrspa (2, 1) | Store the contents of array[2] into rsp[1] |
| 05 | return | Return response, and wait for command |
| 06 | getcmd (0, 0) | Store the value of cmd[0] into array[0] |
| 07 | random (1, 2) | Store a random number (0 to 2) into array[1] |
| 08 | cmpa (3, 0, 1) | Compare array[0] and array[1] contents and store the result into array[3] |
| 09 | setrspa (1, 0) | Store the contents of array[1] into rsp[0] |
| 0a | setrspa (2, 1) | Store the contents of array[2] into rsp[1] |
| 0b | return | Return response, and wait for command |
| 0c | getcmd (0, 0) | Store the value of cmd[0] into array[0] |
| 0d | random (1, 2) | Store a random number (0 to 2) into array[1] |
| 0e | cmpa (4, 0, 1) | Compare array[0] and array[1] contents and store the result into array[4] |
| 0f | setrspa (1, 0) | Store the contents of array[1] into rsp[0] |
| 10 | setrspa (2, 1) | Store the contents of array[2] into rsp[1] |
| 11 | setrspv (1, 2) | Store a value of 1 (to indicate the last) into rsp[1] |
| 12 | ne (2, 0 14) | Jump to line 0×14 if the contents of array[2]≠0 |
| 13 | set (5, 20) | Store 20 (points won per game) into array[5] |
| 14 | ne (3, 0, 16) | Jump to line 0×16 if the contents of array[3]≠0 |
| 15 | addv (5, 5, 20) | Add 20 (points won per game) to array[5] |
| 16 | ne (4, 0, 16) | Jump to line 0×18 if the contents of array[4]≠0 |
| 17 | addv (5, 5, 20) | Add 20 (points won per game) to array[5] |
| 18 | pointv (5) | Add the contents of array[5] to points |
| 19 | setrspa (5, 2) | Store the contents of array[5] (final points) into rsp[2] |
| 1a | end | Return response, and exit |

FIG. 17

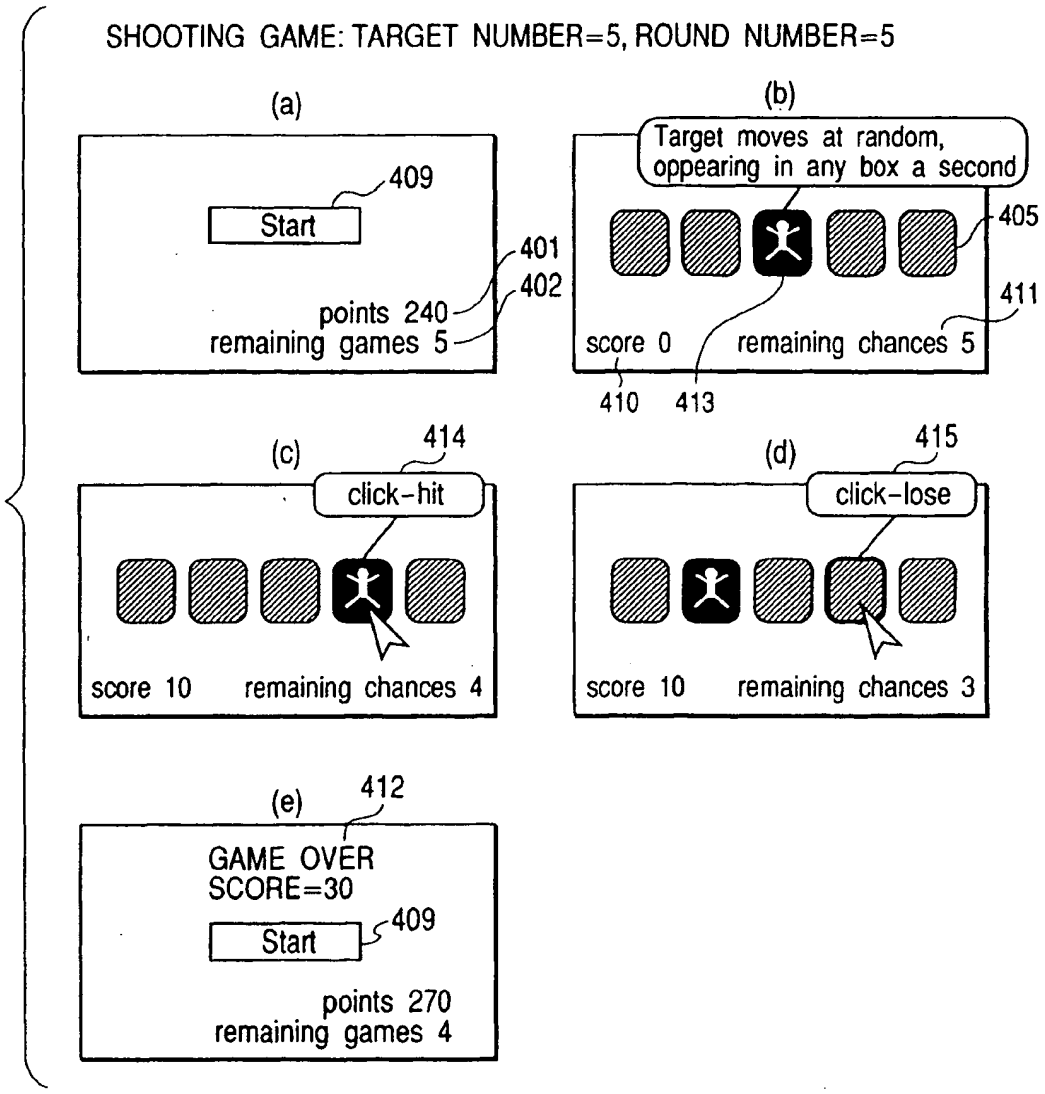
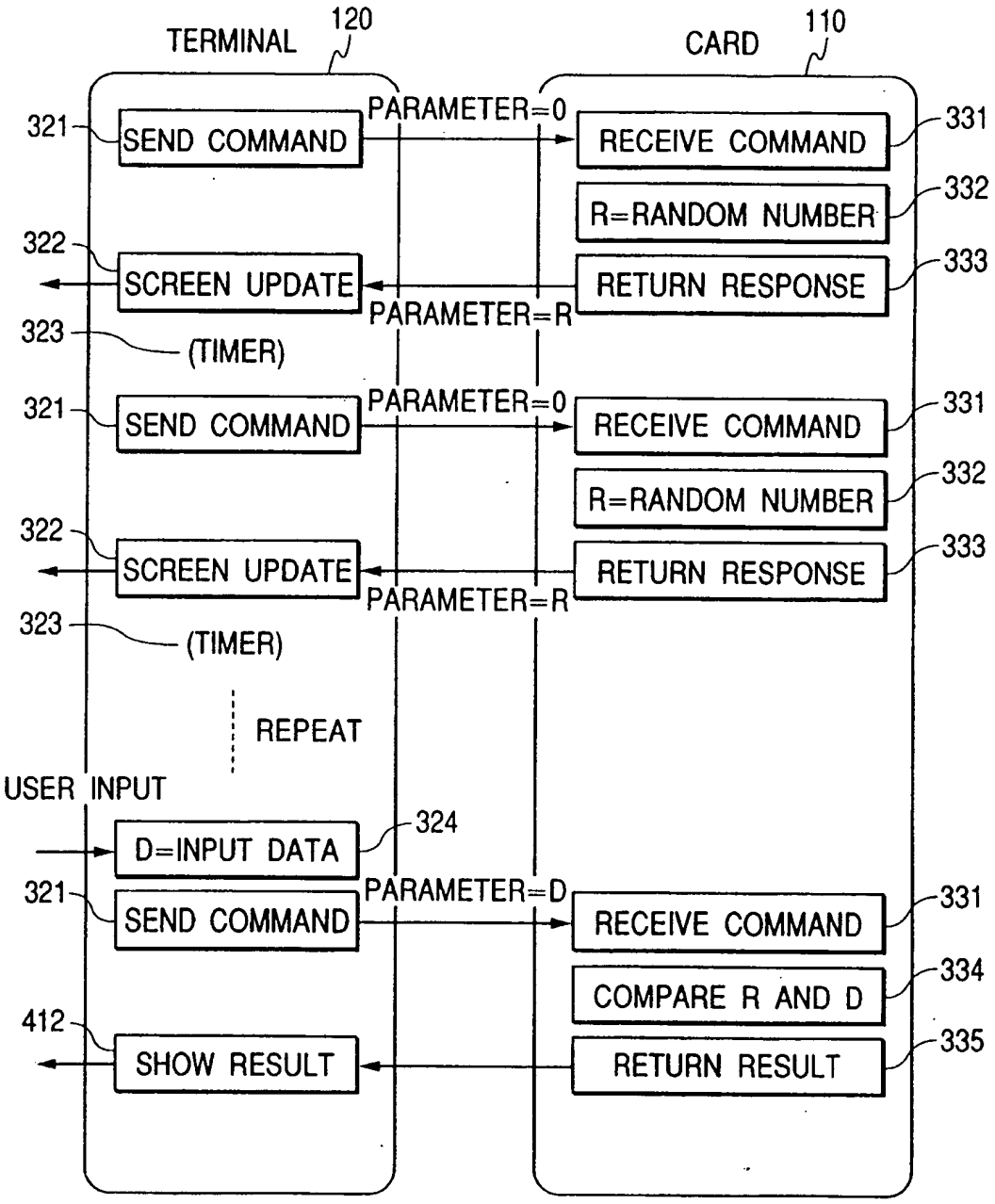

SHOOTING GAME: TARGET NUMBER=5, ROUND NUMBER=5
(a)
409
Start
401
402
points 240
remaining games 5
(b)
Target moves at random, oppearing in any box a second
405
411
score 0
remaining chances 5
410
413
(c)
414
click-hit
score 10
remaining chances 4
(d)
415
click-lose
score 10
remaining chances 3
(e)
412
GAME OVER
SCORE=30
409
Start
points 270
remaining games 4

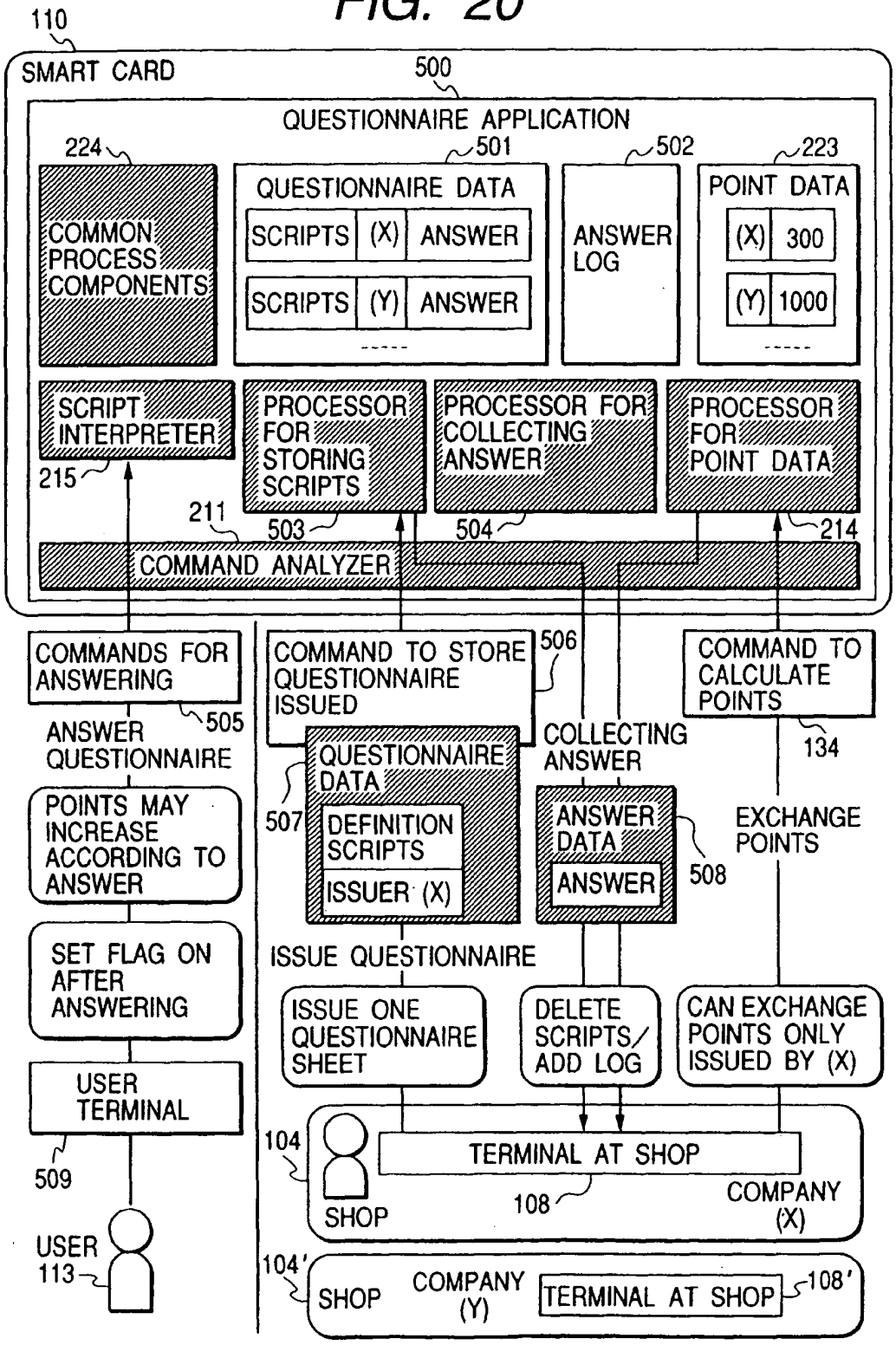
FIG. 18


TERMINAL
120
CARD
110
321
SEND COMMAND
PARAMETER=0
RECEIVE COMMAND
331
R=RANDOM NUMBER
332
322
SCREEN UPDATE
RETURN RESPONSE
333
PARAMETER=R
323
(TIMER)
321
SEND COMMAND
PARAMETER=0
RECEIVE COMMAND
331
R=RANDOM NUMBER
332
322
SCREEN UPDATE
RETURN RESPONSE
333
PARAMETER=R
323
(TIMER)
REPEAT
USER INPUT
D=INPUT DATA
324
321
SEND COMMAND
PARAMETER=D
RECEIVE COMMAND
331
COMPARE R AND D
334
412
SHOW RESULT
RETURN RESULT
335

# FIG. 19

| LINE | SCRIPT | COMMENT |
|---|---|---|
| 00 | set (0, 0) | Store 0 into array[0] (loop index) |
| 01 | set (1, 0) | Store 0 into array[1] (points won) |
| 02 | set (2, 0,) | Store 0 into array[2] (random number to be compared with input) |
| 03 | getcmd (3, 0) | Store the value of cmd[0] into array[3] |
| 04 | eq (3, 0, 12) | Jump to line 0×12 if array[3] contents 0 (Comparing values follows) |
| 05 | cmpa (4, 2, 3) | Compare array[2] and array[3] contents and store the result into array[4] |
| 06 | setrspa (4, 0) | Store the contents of array[4] (result of comparison) into rsp[0] |
| 07 | addv (0, 0, 1) | Increment the counter value contained in array[0] by one |
| 08 | setrspa (0, 1) | Store the contents of array[0] (loop count) into rsp[1] |
| 09 | ne (4, 0, 0b) | Jump to line 0×0b if the contents of array[4]≠0 (means loosing) |
| 0a | addv (1, 1, 10) | Add 10 (additional points) to array[1] |
| 0b | eq (0, 5, 0e) | Jump to line 0×0e if the contents of array[0]=5 (last) |
| 0c | return | Return response and wait for command |
| 0d | jump (03) | Jump to line 0×03 (beginning of loop) (Last loop execution follows) |
| 0e | pointv (1) | Add the contents of array[1] to points |
| 0f | setrspa (1, 2) | Store the contents of array[1] (points) into rsp[2] |
| 10 | setrspv (1, 3) | Store a value of 1 (to indicate the last) into rsp[3] |
| 11 | end | Return response and exit (Generating a random number follows) |
| 12 | random (2, 5) | Store a random number (0 to 4) into array[2] |
| 13 | addv (2, 2, 1) | Increment the value contained in array[2] by one (the range of random numbers change to 1 to 5) |
| 14 | setrspa (2, 0) | Store the contents of array[1] (random number) into rsp[0] |
| 15 | return | Return response and wait for command |
| 16 | jump (0, 3) | Jump to line 0×03 (beginning of loop) |

FIG. 20


110
SMART CARD
500
QUESTIONNAIRE APPLICATION
224
501
502
223
COMMON PROCESS COMPONENTS
QUESTIONNAIRE DATA
SCRIPTS (X) ANSWER
SCRIPTS (Y) ANSWER
ANSWER LOG
POINT DATA
(X) 300
(Y) 1000
SCRIPT INTERPRETER
215
PROCESSOR FOR STORING SCRIPTS
PROCESSOR FOR COLLECTING ANSWER
PROCESSOR FOR POINT DATA
211
503
504
214
COMMAND ANALYZER
COMMANDS FOR ANSWERING
505
ANSWER QUESTIONNAIRE
POINTS MAY INCREASE ACCORDING TO ANSWER
SET FLAG ON AFTER ANSWERING
USER TERMINAL
509
USER
113
COMMAND TO STORE QUESTIONNAIRE ISSUED
506
QUESTIONNAIRE DATA
507
DEFINITION SCRIPTS
ISSUER (X)
ISSUE QUESTIONNAIRE
COLLECTING ANSWER
ANSWER DATA
ANSWER
508
COMMAND TO CALCULATE POINTS
134
EXCHANGE POINTS
ISSUE ONE QUESTIONNAIRE SHEET
DELETE SCRIPTS/ ADD LOG
CAN EXCHANGE POINTS ONLY ISSUED BY (X)
104
SHOP
TERMINAL AT SHOP
108
COMPANY (X)
104'
SHOP
COMPANY (Y)
TERMINAL AT SHOP
108'